AD-A243 775

SBI/ NORDA

(2)

Ocean Simulation Model - Version 2 First Order Frontal Simulation

K. D. Saunders
Oceanography Division
Ocean Science Directorate

C. F. Cheng
Sverdrup Technology Corporation

 Naval Oceanographic and Atmospheric Research Laboratory, Stennis Space Center, Mississippi 39529-5004.

ABSTRACT

Temperature, salinity, density, and sound velocity are the properties of most interest to the physical oceanographer and acoustician. The Naval Oceanographic and Atmospheric Research Laboratory has produced a first-level numerical simulation model that can produce simulated sections of temperature, salinity, density and sound velocity in the vicinity of an oceanic front. The user controls the definition of the front.

This technical note documents the algorithms used in the simulation model and provides a users guide to the programs. Two programs are documented. The first program generates the front, and the second produces plots of the frontal properties.

ACKNOWLEDGMENTS

We would like to thank Drs. A.W. Green, M. Briscoe, P. Smith and our other colleagues for their generous advice and Ms. Jaime Ratliff for editing this technical note. This work was funded by the Office of Naval Technology, Program Element 602435N, CDR Lee Bounds, Program Manager.

CONTENTS

OCEAN SIMULATION MODEL - VERSION 2
FIRST ORDER FRONTAL LOCATION SIMULATION

INTRODUCTION

There has long been a need for providing realistic simulations of frontal properties in the open ocean. These properties include temperature, salinity, density, sound velocity, chemical and biological tracers. The computer program documented in this technical note is the first step in the Naval Oceanographic and Atmospheric Research Laboratory's exploratory development simulation program to provide this capability to the Navy and academic research community.

This program was developed using the following principles:

1. the location of the front in the horizontal plane is first obtained,
2. the depth of the center of the pycnocline is then computed,
3. the three dimensional density field is then determined, and
4. the fields of temperature, salinity, and tracers.

The algorithms for the above steps are first discussed, followed by a technical description of the program, a users guide and finally, full listings of the associated program components.

ALGORITHMS

ALGORITHM FOR THE FRONTAL LOCATION IN THE HORIZONTAL PLANE

The algorithm for the first order front simulation is based on the physical observation that disturbances along a front grow and propagate in the local curvilinear coordinates of the front. We simulate this behavior by creating the front as a series of approximations, beginning with a sine wave with growing downstream amplitude. For each subsequent approximation, a higher wave number sine wave is added to the previous approximation in coordinates along and normal to the front, as illustrated in the figure below.

The alongstream coordinate is denoted by ξ and the cross-stream, or normal coordinate, is denoted by η.

The new position of the front is given by

$$y^{n+1}(\xi) = y^{n}(\xi) + \delta y(\xi, \eta^{n}[\xi]) \quad , \tag{1}$$

$$x^{n+1}(\xi) = x^{n}(\xi) + \delta x(\xi, \eta^{n}[\xi]) \quad . \tag{2}$$

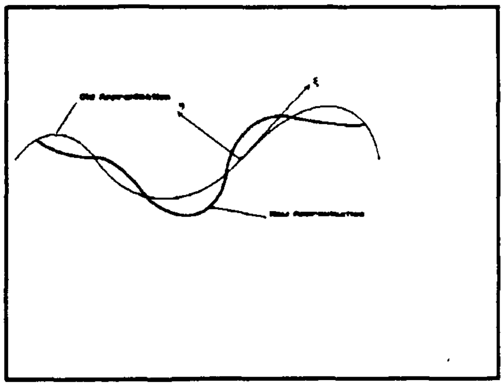

Figure 1 Geometry of front approximation.

An example of this method is given in figure 2, below.

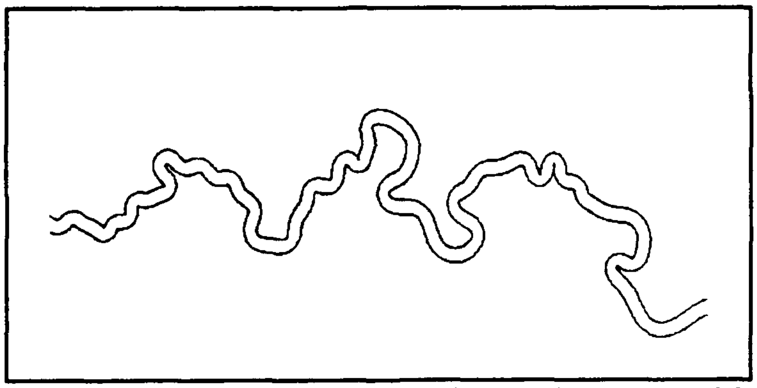

Figure 2 Example of a front simulated by the first level model.

The details of the algorithm are completed by specifying $\eta^{(n)}(\xi)$:

$$\eta^{n}(\xi) = a(k_n)[\, 1 + b\, \xi \,]\, \sin(k_n \xi + \phi_n) \;, \tag{3}$$

where

$a(k_n) = S^{1/2}(k_n)$,

$S(k)$ = wavenumber spectrum of the front,

b = streamwise growth factor,

k_n = n-th wave number,

ϕ_n = random phase of the n-th wave component,

ξ = alongstream coordinate.

ALGORITHM FOR THE THERMOCLINE DEPTH

The depth of the thermocline (defined as the depth of the maximum Brunt-Väisälä frequency profile) can then be computed from the relation:

$$\nabla^2 h - \delta_k^2 h = -\delta_k^2 h_k \, , \tag{4}$$

where

δ_k^{-1} = the Rossby radius of deformation = ND_k/f,

N = the maximum Brunt-Väisälä frequency,

f = the Coriolis parameter,

D_k = the depth of the appropriate isopycnal (on the boundary where $h=h_k$).

The boundary conditions are given as $h = 0$ along $x(\xi,\eta)$, $y(\xi,\eta)$ and $h \to h_+$ as $y \to +\infty$ and $h \to h_-$ as $y \to -\infty$, and

h_k = h_+ when the point is on the positive side of the front,

= h_- when the point is on the negative side of the front.

This equation can be easily solved by successive over-relaxation on a given grid. The primary problem in implementing the solution is determining whether a point is on the positive or negative side of the front.

Once the field of h(x,y) is determined, the rest of the density field can be filled in using the GDEM profile scheme. If a mixing of water mass properties is desired, the TS curve can be filled in as a mixture, the proportions being determined by the value of h.

ALGORITHM FOR COMPUTING THE DENSITY, TEMPERATURE, AND SALINITY FIELDS

This algorithm is based on the assumption of a constant form for the BV profile, advected vertically according to the depth of the maximum. The resultant temperature and salinities are then computed by assuming temperature and salinity mix along surfaces of constant density, the amount of mixing determined by the relative vertical displacement of the isopycnal from a reference level.

1. Locate desired Latitude and Longitude of the frontal region and points on either side of the front.

2. Get $T_0(z)$ and $S_0(z)$ profiles from appropriate data base (Use Levitus 5°).

3. Compute $\rho_0(z)$ and $N_0(z)$.

4. Compute z_m by the relation $N_{max} = \max[N_0(z)] = N_0(z_m)$.

5. Compute the Rossby radius of Deformation using N_{max}.

6. Compute h(x,y) field from the depth algorithm as previously described.

7. Compute N(x,y,z) by displacing $N_0(z)$ by h(x,y), i.e. $N(x,y,z) = N_0(z + h(x,y))$, truncating for $z > 0$ and extending $N_0(z + h) = N_0(-H)$ when $z + h < -H$.

8. Compute new density field $\rho(x,y,z)$ by integrating N(x,y,z) from $\rho(x,y,h=0) = \rho_0$ up down at each x,y point.

9. Let $T^+(\rho)$, $S^+(\rho)$ correspond to T, S as $h \rightarrow h^+$ and $T^-(\rho)$, $S^-(\rho)$, correspond to T, S as $h \rightarrow h_-$. Then, compute $T(x,y,z) = \{(h_+ - h(x,y))^\alpha T^-(\rho[x,y,z]) + (h(x,y) - h_-)^\alpha T^+(\rho[x,y,z])\}/\{(h_+ - h)^\alpha + (h - h_-)^\alpha\}$ and similarly for S.

10. Use the values of T,S to check consistency with N(x,y,z) and compute sound speed profiles.

DETAILED ALGORITHM FOR T/S "BLENDING" ALONG ISOPYCNALS

(Expansion of Item 9 in previous algorithm)

1. Let the user input two sets of (Latitude/Longitude) pairs: (θ_i, λ_i) ,i = 1,2.

2. Retrieve the temperature and salinity profiles corresponding to these locations, $T_i(z) = T(\theta_i, \lambda_i, z)$, i = 1,2.

3. Compute the field of $\sigma(r,z)$ and h(r) where r is the range coordinate along the desired section. [Here, $h(r=0)$ corresponds to location 1 and is the maximum value of h and $h(r=r_{max})$ corresponds to location 2 and is the minimum value of h.]

4. Determine the range of σ over the section, i.e. find , $\sigma_{max} = \max(\sigma(r,z)\ \forall\ r,z \in \text{section})$ and $\sigma_{min} = \min(\sigma(r,z)\ \forall\ r,z \in \text{section})$.

5. Compute $T_i(\sigma)$, $i = 1,2$ for $\sigma \in [\sigma_{min},\sigma_{max}]$ at uniformly spaced $\Delta\sigma$. In order to interpolate the temperatures beyond the σ limits at locations 1 and 2, extend the temperatures and salinities by padding with the end values of z and extend σ by a linear extrapolation from the ends. [That is at $i = 1$ $(r = 0)$, $\sigma(0,z) = \sigma(0,0) + \partial_z\sigma(0,0)z$ for $z < 0$ and $\sigma(0,z) = \sigma(0,4000) + \partial_z\sigma(0,4000)(z-4000)$ for $z > 4000$ m, and similarly for $i = 2$ $(r = r_{max})$.]

6. Compute $T(r,\sigma)$ by the following method:

$$T(r,\sigma) = \frac{(h_1 - h(r))^{\alpha}\, T_2(\sigma) + (h(r) - h_2)^{\alpha}\, T_1(\sigma)}{(h_1 - h(r))^{\alpha} + (h(r) - h_2)^{\alpha}} \quad (5)$$

   where α is a user selected exponent. (Default = 1).

7. From the relation $\sigma = \sigma(r,z)$, compute $z = z(r,\sigma)$. Use this relation to map $T(r,\sigma) \rightarrow T(r,z)$.

8. Repeat (or compute concurrently) the processes in 5-7 for salinity to get the field of $S(r,z)$.

8a. A new field of $\sigma(r,z)$ is recomputed from the blended $T(r,z)$ and $S(r,z)$ fields and it is ensured that σ is a nondecreasing function of depth (i.e., stable stratification everywhere). If it is not, the depths are sorted to ensure increasing σ and the temperatures and salinities carried along with the sort. This checking is optional at the user's discretion.

9. Compute the field of sound velocity $c(r,z)$.

PROGRAM FSM - FRONT SIMULATION MODELING

OPERATING INSTRUCTIONS:

Control File

Upon entry into program FSM, the user is prompted to enter a control file (CF) name. The control file specified by the user will be checked for existence. A new control file will be allocated if the file does exist. All the processing parameters stored in the control file are also initialized to default values. This control file is used to store input and output file names, processing parameters and other data output by program FSM, such as: the x,y coordinates of the front curve generated by directive FRNT, the computed temperature, salinity, density, and frequency profiles generated by directive TS, etc. The structure of the control file is described in detail later in this section.

Directives: Overview

Program FSM includes 13 directives. Once the name of the control file is entered, the user may execute any of these directives. Directive LD lists the names and the description of the directives. Directive SF allows the user to set the names for the files required by directives such as: HELM, TS, SIG and SV to use as input and output. Directive IN initializes all the processing parameters required by directives such as: FRNT, HELM, TS, and SV to start the operations. Directive SP allows the user to set processing parameters to desired values.

Generating the Front Position

Directive FRNT uses the current clock time as initial seed to call the intrinsic random number generator (RAN) to generate a front curve. The number of points for the front curve is defined by the parameter NPTS. The x,y coordinates of the points constituting the front curve are stored in the control file.

Thermocline Depth Generation

Once the front curve is generated, the user may execute directive HELM to start the Helmholtz equation solver process. This directive first rescales the x,y points of the front curve to fit in a 640 x 480 gridded space. Value 1 is assigned to all the pixels above the curve; value -1 is assigned to all the pixels below the curve and value 0 is assigned to the pixels along the curve. This gridded matrix is also stored in the control file and used as a mask for the directive to perform the HELM computation process. The final output from the HELM process is stored into an external file. The internal usage name for this file is named HMF. The HELM process actually involves iterative computations. The number of iterations required is defined by the parameter ITER. Therefore, the process can potentially be very time consuming if the parameter ITER is set to a large number. Directive RES was designed to allow the user to resume the HELM operation from where it was terminated previously.

Temperature/Salinity Field Generation

Before directive TS is executed, the user should use directive SF to specify the name and the location of the Levitus database file (DBF) as input. In order to read the temperature and salinity data for the desired location from the DBF file, the parameters LAT(itude) and LONG(itude) should also be set previously by directive SP. After the temperature and salinity data are read in from the DBF file, directive TS will interpolate these two sets of data with evenly spaced depths ranging from 0 to 4000 m. The depth increment is defined by the parameter IZ and must be evenly divisible by the maximum depth 4000 m. After the interpolation is done, the density and frequency values at these depth locations are computed. The data, including depth, temperature, salinity, density and frequency values, are then stored into the control file.

Density Field Recomputation

Directive SIG requires the output from directives HELM and TS as input. In addition, the user is required to specify two end points defining the section of interest in the HMF file to compute a new integrated sigma (density) field. The output, sigma field, is stored into an external file with usage name SFF.

Sound Velocity Field Generation

Directive SV requires the output from the directives HELM and SIG as input. The user is also required to specify two sets of latitude and longitude for the directive to read the temperature and salinity data from the DBF file. Based upon the input data, directive SV computes blended temperature and salinity values. The blended temperature and salinity data may then be used to compute sound velocity values. The directive SV allows the user to select at least one of these computed data sets, blended temperature, blended salinity, and sound velocity, to be output to an external file. The usage names for these three output files are TF, SF, and SVF, respectively. The user should use the directive SF to selectively define these output file names before this directive SV is executed. Notice that all the files output by program FSM including: QMF, SSF, TF, SF, and SVF are checked for existence.

Program FSM will create a new file and write the output to the file if it does not exist, otherwise program FSM will write the output over to the existing file.

FSM DIRECTIVES

DIRECTIVE **DESCRIPTION**

LD - List program main directives

SF - Set/list external files to input/output

IN - Initialize processing parameters to default values

SP - Set processing parameters

FRNT - Generate front curve

LIST - List x,y coordinates of front curve

HELM - Apply Helmholtz equation solver

RES - Resume Helmholtz equation solver

TS - Generate temperature, salinity, density, and frequency profiles

LP - List temperature, salinity, density, and frequency profiles

SIG - Generate integrated sigma field

SV - Generate sound velocity, blended temperature, or blended salinity output

END - End the program

FSM PARAMETERS

PARAMETERS **DESCRIPTION**

The following parameters are required by the directive FRNT:

NPTS - Number of points used for front curve (default to 5000)

LM - Number of points (+/-) to sample (default to 10)

RLP - Ripple power (default to 2.0)

ITER - Number of iterations to generate front curve (default to 20)

The following parameters are required by the directive HELM:

ALFA - Relaxation coefficient (default to 1.7)

BETA - Rossby Deformation Radius (default to 20.0)

H - H grid spacing (default to 0.4)

MAXH - Maximum H value (default to 100)

MINH - Minimum H value (default to -100)

JTER - Number of iterations for HELM solver (default to 1000)

The following parameters are required by the directive TS:

LAT - Latitude of desired location in degrees (default to 30)

LONG - Longitude of desired location in degrees (default to -70)

IZ - Depth increment (default to 10)

The following parameters are required by the directive SV:

E - Exponent variable (default to 1)

RCF - Flag indicating necessity of recomputing sigma field based upon blended temperature and salinity (default is NO)

FSM DESCRIPTION AND FORMAT OF CONTROL FILE (CF)

The control file used by this program is a random access file. Each record of the file is 400 bytes long. The following items are stored in the control file:

. Processing parameters
. External input/output file names and last written data and time
. The floating-point x,y coordinates of the points constituting the front curve
. A 640 x 480 16-bit gridded matrix storing the mask of the front curve
. Computed floating-point temperature, salinity, density, and frequency profiles at a specific latitude and longitude location

The format of the first two records of the control file is shown below:

RECORD	WORD	TYPE	DESCRIPTION
1	1	INTEGER*4	Control file identifier (56789)
	2	"	Number of records in the file
	3	"	Beginning record at which the x,y coordinates of the front curve is stored
	4	"	Not used
	5	"	Beginning record at which the mask of the front curve is stored
	6	"	Not used
	7	"	Beginning record at which the temperature, salinity, density and frequency profiles are stored
	8-10	"	Not used
	11	"	Parameter NPTS
	12	"	Parameter LM
	13	REAL	Parameter RLP
	14	INTEGER*4	Parameter ITER
	15-19	"	Not used
	20	"	Number of points of the front curve
	21	REAL	Parameter ALFA
	22	"	Parameter BETA
	23	"	Parameter H
	24	INTEGER*4	Parameter MINH
	25	"	Parameter MAXH
	26	"	Parameter JTER
	27	"	Not used
	28	"	Next iteration number
	29	"	Location at which the front curve crosses left boundary of the mask
	30	"	Location at which the front curve crosses right boundary of the mask

	31	"	Parameter LAT
	32	"	Parameter LONG
	33	"	Parameter IZ
	34	"	Not used
	35	"	MAXH used by directive TS
	36	"	MINH used by directive TS
	37	"	IZ used by directive TS
	38	"	Number of evenly spaced depths
	39	"	Location at which the maximum frequency value is located
	40	REAL	Sigma value corresponding to the location of the maximum frequency
	41	INTEGER*4	X coordinate of the first end point
	42	"	Y coordinate of the first end point
	43	"	X coordinate of the second end point
	44	"	Y coordinate of the second end point
	45	REAL	Minimum sigma value
	46	"	Maximum sigma value
	47	INTEGER*4	MINH used by directive SIG
	48	"	MAXH used by directive SIG
	49	"	IZ used by directive SIG
	50	"	Number of evenly spaced depths
	51	REAL	Parameter E
	52	INTEGER*4	Parameter RCF
	53-100		Not used
2	1-12	Integer*4	DBF file name
	21-26	"	HMF file name
	27-28	"	HMF file last written date
	29-30	"	HMF file last written time
	31-36	"	SFF file name
	37-38	"	SFF file last written date
	39-40	"	SFF file last written time
	41-46	"	SVF file name
	47-48	"	SVF file last written date
	49-50	"	SVF file last written time
	51-56	"	TF file name
	57-58	"	TF file last written date
	59-60	"	TF file last written time
	61-66	"	SF file name
	67-68	"	SF file last written date
	69-70	"	SF file last written time
	71-100		Not used

FSM INPUT/OUTPUT FILES

DBF: Levitus data base file name required by directives TS and SV as input (default to [kim.modelocean.database]levitus.dat).
File type: random access.
Record length: 720 bytes.

HMF: HELM solver file output by directives HELM and RES.
Data type: single-precision output
File type: random access.
Record length: 1920 bytes.

SFF: SIGMA field file output by directive SIG
Data type: single-precision output
File type: random access.
Record length: ((maximum depth 4000/parameter IZ)+1)*4 bytes

SVF: Sound velocity file output by directive SV.
Data type: single-precision output
File type: random access.
Record length: ((maximum depth 4000/parameter IZ)+1)*4 bytes

TF: Blended temperature file output by directive SV.
Data type: single-precision output
File type: random access.
Record length: ((maximum depth 4000/parameter IZ)+1)*4 bytes

SF: Blended salinity file output by directive SV.
Data type: single-precision output
File type: random access.
Record length: ((maximum depth 4000/parameter IZ)+1)*4 bytes

SUBROUTINES/FUNCTIONS CALLED BY THE MAIN PROGRAM (FSM)

Subroutine CKFILE	-	Checks existence and record size of CF, DBF, HMF, SFF, SVF, TF or SF file
Subroutine CLSFIL	-	Closes DBF, HMF, SFF, SVF, TF and SF files
Subroutine INPAR	-	Initializes processing parameters
Subroutine SETFS	-	Sets file name for DBF, HMF, SFF, SVF, TF or SF file
Subroutine RUI	-	Reads user's input as character string and parses it into Hollerith substrings, integer and floating-point values.
Subroutine SETPAR	-	Sets processing parameters
Subroutine RDTS	-	Reads temperature and salinity profiles from the DBF file based upon give latitude and longitude location
Subroutine DLXY	-	Linearly interpolates integer x,y locations between two end points
Subroutine SIGINT	-	Integrates sigma values
Subroutine LINTPL	-	Linearly interpolates y values within input given x limits and linearly y values outside x limits.
Subroutine DATIME	-	Gets current clock time by calling system dependent routines: IDATE and TIME
Subroutine INTRPL	-	Akima interpolation routine
Function BVFRQ	-	Computes Brunt-Väisälä frequency
Function SVAN	-	Computes sigma value based upon given temperature, salinity, pressure and depth
Function THETA	-	Computes potential temperature
Function ATG	-	Computes temperature gradient [°C per decibar]
Function SVEL	-	Computes sound velocity based upon given temperature, salinity and depth

PROGRAM FSP - FRONTAL PLOTTING PROGRAM

FSP OPERATING INSTRUCTIONS:

Program FSP uses the Display Integrated Software System and Plotting Language (DISSPLA) to plot a HMF, SFF, SVF, TF, or SF file output from the program FSM to an special file named POPFIL.DAT. After the plot is generated, the user may invoke the postprocessor program DISSPLA or DISS105 to physically plot the data stored in POPFIL.DAT to an external device such as a graphic terminal or a plotter. For the description and the format of the files to be plotted, the user may refer to the documentation for program FSM.

Initial Prompts

Upon entry into program FSP, the user will be prompted for entering the name of a control file (CF). This control file should be the file previously created by program FSM and should contain the names of the files (HMF, SFF, SVF, TF, and SF) to be plotted, otherwise the program will be terminated.

Once the control file is checked, program FSP will automatically put the user on the file selection mode. The user should then select the file to be plotted from one of the existing HMF, SFF, SVF, TF, and SF files.

Plotting Parameters

There are six plotting parameters. Parameters WID and HT define the width and height (X and Y axes) of the plot. Parameters IC, LC, IR, and LR define an area of interest within the selected file to be plotted. After a file being selected, these parameters are automatically set to default values. Parameters WID and HT are set to the maximum of 14 and 11 inches, respectively. Parameters IC, LC, IR, and LR (defining the initial and last columns and rows of the area of interest are set to the limits of the selected file. Notice that the columns and rows of the area of interest are designated as the Y and X axes of the output plot, respectively.

Directives

Program FSP includes six directives. Directive LD lists the names and description of the directives. Directive IN resets the plotting parameters to the default values mentioned above. Directive SP allows the user to change these plotting parameters to desired values. Directive SF puts the user back to the file selection mode such that a new file may be selected for plotting. Directive PLOT calls DISSPLA routine CONMAK to generate contour data and output it to file POPFIL.DAT. After this directive is invoked, the minimum and the maximum data values within the area of interest will be displayed. The user is then asked to enter a constant increment value (INCR) for generating contour lines. Each contour line generated will contain the same data value. Starting from the minimum value, the data value for each contour line is an increment of INCR. The user may refer to the DISSPLA users manual for more detailed information about the technique used by routine CONMAK for generating contour lines.

Contouring Errors

The user may consider to set the area of interest to a smaller area or set INCR to a large value to re-execute directive PLOT if the error message "TOO MUCH DATA PASSES TO ROUTINE CONMAK" is encountered during the execution of this directive.

FSP DIRECTIVES

DIRECTIVE		DESCRIPTION
LD	-	List program main directives
IN	-	Initialize plotting parameters to default values
SP	-	Set plotting parameters
PLOT	-	Generate a plot file named POPFIL.DAT
SF	-	Return to file selection mode
END	-	End the program

FSP PARAMETERS

PARAMETERS		DESCRIPTION
WID	-	Width of plot (default to 14 inches)
HT	-	Height of plot (default to 11 inches)
IC	-	Initial column of the file to be plotted (default to 1)
LC	-	Last column of the file to be plotted (default to record size of selected file)
IR	-	Initial row of the file to be plotted (default to 1)
LR	-	Last row of the file to be plotted (default to last record of selected file)

INPUT FILES

See the documentation for program FSM for more detailed description about HMF, SFF, SVF, TF, and SV files.

SUBROUTINES/FUNCTIONS CALLED BY THE MAIN PROGRAM (FSP)

Subroutine CKFILE	-	Checks existence and record size of CF, DBF, HMF, SFF, SVF, TF, or SF file
Subroutine CLSFIL	-	Closes DBF, HMF, SFF, SVF, TF, and SF files
Subroutine RUI	-	Reads user's input as character string and parses it into Hollerith, substrings, integer and floating-point values.
Subroutine SELECF	-	Selects a file to plot
Subroutine SETPAR	-	Sets plotting parameters
Subroutine DLXY	-	Linearly interpolates integer x,y locations between two end points
Subroutine DATIME	-	Get current clock time by calling system dependent routines: IDATE and TIME

The following subroutines are called from DISSPLA software package:

COMPRS, PAGE, AREA2D, XNAME, YNAME, HEADIN, GRAF, BCOMON, CONMAK, CONLIN, CONTUR, ENDPL and DONEPL

ALGORITHM

None

FSM PROGRAM LISTING

```
C
C*****************************************************************************
C PROGRAM NAME:FSM.FOR - FRONT SIMULATION MODELING
C DATE: JULY 2, 1990
C PROGRAMMER: TIGER CHENG (SVERDRUP)
C*****************************************************************************
C
      PROGRAM FSM
      PARAMETER (MXFCPS=5000)
      PARAMETER (MXCOLS=640,MXROWS=480,MXWS=MXCOLS*MXROWS)
      PARAMETER (NBPRCF=400,NWPRCF=NBPRCF/4)
      PARAMETER (NDIRS=13)
      PARAMETER (NWFN=6)
      PARAMETER (MXZS=30)
      PARAMETER (MXFS=7)
      PARAMETER (MXVS=MXROWS+MXCOLS)
      INTEGER*2 MASK(MXCOLS,MXROWS)
      INTEGER*4 BLANKS,YES(2),IH(20),IA(10),IBUF(NWPRCF*2),JBUF(NWPRCF)
      INTEGER*4 IDIRS(NDIRS),IX(-10:MXFCPS+10),IY(-10:MXFCPS+10),
     *          MASK4(MXWS/2),IXS(MXVS),IYS(MXVS),INDEX(MXVS+MXVS),
     *          FILTYP(MXFS),LUS(MXFS),LUSO(MXFS)
      REAL XDBF(180),XBUF(NWPRCF*2),ZLEV(MXZS),FA(10)
      REAL DEP(MXVS),TEMP(MXVS),SAL(MXVS),SIG(MXVS),BVF(MXVS),
     *     DEP2(MXVS),TEMP2(MXVS),SAL2(MXVS),SIG2(MXVS),BVF2(MXVS),
     *     DEPO(MXVS),TEMPO(MXVS),SALO(MXVS),SIGO(MXVS),BVFO(MXVS),
     *     DEPO2(MXVS),TEMPO2(MXVS),SALO2(MXVS),SIGO2(MXVS),
     *     BVFO2(MXVS),SV(MXVS)
      REAL X(MXFCPS),Y(MXFCPS),T(MXFCPS),
     *     X1(-10:MXFCPS+10),Y1(-10:MXFCPS+10)
      REAL DBUF(NWPRCF),AK(0:100),A(0:100),PHI(0:100)
      REAL Q(MXCOLS,MXROWS),QQ(MXWS),ETA(-1:1)
      CHARACTER*24 CFN,HMFN,SFFN,SVFN,TFN,SFN
      CHARACTER*48 DBFN,DFN
      EQUIVALENCE (LUS(1),LUCF),(LUS(2),LUHMF),(LUS(3),LUDBF),
     *            (LUS(4),LUSFF),(LUS(5),LUSVF),(LUS(6),LUTF),
     *            (LUS(7),LUSF)
      EQUIVALENCE (IBUF(1),XBUF(1),DBUF(1)),(IBUF(NWPRCF+1),JBUF(1))
      EQUIVALENCE (Q(1,1),QQ(1),T(1),XDBF(1)),
     *            (QQ(MXFCPS+1),X1(-10),IX(-10)),
     *            (QQ(MXFCPS*2+22),Y1(-10),IY(-10)),
     *            (QQ(MXFCPS*3+43),X(1)),(QQ(MXFCPS*4+43),Y(1)),
     *            (QQ(MXFCPS*5+43),AK(0)),(QQ(MXFCPS*5+154),A(0)),
     *            (QQ(MXFCPS*5+255),PHI(0))
      EQUIVALENCE (MASK(1,1),MASK4(1),IXS(1)),(MASK4(MXVS+1),IYS(1)),
     *            (MASK4(MXVS*2+1),DEP(1)),(MASK4(MXVS*3+1),TEMP(1)),
     *            (MASK4(MXVS*4+1),SAL(1)),(MASK4(MXVS*5+1),SIG(1)),
     *            (MASK4(MXVS*6+1),BVF(1)),(MASK4(MXVS*7+1),DEP2(1)),
     *            (MASK4(MXVS*8+1),TEMP2(1)),(MASK4(MXVS*9+1),SAL2(1)),
     *            (MASK4(MXVS*10+1),SIG2(1)),(MASK4(MXVS*11+1),BVF2(1)),
     *            (MASK4(MXVS*12+1),DEPO(1)),
     *            (MASK4(MXVS*13+1),TEMPO(1)),
     *            (MASK4(MXVS*14+1),SALO(1)),(MASK4(MXVS*15+1),SIGO(1)),
     *            (MASK4(MXVS*16+1),BVFO(1)),
     *            (MASK4(MXVS*17+1),DEPO2(1)),
     *            (MASK4(MXVS*18+1),TEMPO2(1)),
     *            (MASK4(MXVS*19+1),SALO2(1)),
     *            (MASK4(MXVS*20+1),SIGO2(1)),
     *            (MASK4(MXVS*21+1),BVFO2(1)),(MASK4(MXVS*22+1),SV(1)),
     *            (MASK4(MXVS*23+1),INDEX(1))
      DATA IRD/5/
      DATA IWR/6/
      DATA IDCF/56789/
      DATA ZLEV/0.,10.,20.,30.,50.,75.,100.,125.,150.,200.,250.,
     *          300.,400.,500.,600.,700.,800.,900.,1000.,1100.,
     *          1200.,1300.,1400.,1500.,1750.,2000.,2500.,3000.,
     *          3500.,4000./
```

```
      DATA BLANKS/4H    /
      DATA IDIRS/4HLD  ,4HSF  ,4HIN  ,4HSP  ,4HFRNT,4HLIST,4HHELM,
     *          4HRES ,4HTS  ,4HLP  ,4HSIG ,4HSV  ,4HEND /
      DATA YES/4HY   ,4HYES /
      DATA FILTYP/4HCF  ,4HHMF ,4HDBF ,4HSFF ,4HSVF ,4HTF  ,4HSF  /
      DATA LUS/3,4,7,8,9,10,11/
      DATA LUSO/7*-1/
      DATA DBFN/'DRB5:[KIM.MODELOCEAN.DATABASE]LEVITUS.DAT      '/
C
C
      WRITE(IWR,10)
10    FORMAT(2X,'FRONT SIMULATION MODELING PROGRAM')
      WRITE(IWR,20)
20    FORMAT(2X,'ENTER CONTROL FILE (CF) NAME')
      CALL RUI(IRD,IWR,IH,FA,IA,NVS)
      IF(IH(1) .EQ. BLANKS) STOP
      WRITE(CFN,30) (IH(I),I=1,NWFN)
30    FORMAT(6A4)
C
C CHECK EXISTENCE OF CONTROL FILE
C
      CALL CKFILE(IWR,2,CFN,1,LUS,LUSO,FILTYP,NWPRCF,IST)
      IF(IST) 40,42,50
40    STOP
C
C CHECK FILE IDENTIFIER FOR EXISTING CONTROL FILE
C
42    READ(LUCF,REC=1) (IBUF(I),I=1,NWPRCF)
      IF(IBUF(1) .NE. IDCF) THEN
         WRITE(IWR,44)
44       FORMAT(2X,'WRONG CONTROL FILE IS USED')
         GO TO 900
      ENDIF
      READ(LUCF,REC=2) (JBUF(I),I=1,NWPRCF)
      GO TO 200
C
C INITIALIZE NEW CONTROL FILE
C
50    IBUF(1)=IDCF
      IBUF(2)=2
      DO 60 I=3,NWPRCF
         IBUF(I)=0
60    CONTINUE
      CALL INPAR(MXFCPS,IBUF)
      WRITE(LUCF,REC=1) (IBUF(I),I=1,NWPRCF)
      DO 70 I=1,NWPRCF
         JBUF(I)=BLANKS
70    CONTINUE
      READ(DBFN,80) (JBUF(I),I=1,12)
80    FORMAT(12A4)
      WRITE(LUCF,REC=2) (JBUF(I),I=1,NWPRCF)
      GO TO 200
C
C USER SELECTS A DIRECTIVE TO PROCESS
C
100   WRITE(IWR,130)
130   FORMAT(2X,'ENTER PROGRAM MAIN DIRECTIVE NAME')
      CALL RUI(IRD,IWR,IH,FA,IA,NVS)
      DO 150 IGO=1,NDIRS
         IF(IH(1) .EQ. IDIRS(IGO)) GO TO 190
150   CONTINUE
      WRITE(IWR,160)
160   FORMAT(2X,'INVALID PROGRAM MAIN DIRECTIVE')
      GO TO 100
C
C
190   GO TO (200,300,400,500,600,700,800,900,1000,1100,1200,1300,
     *       9000),IGO
C
```

```
C LD - LIST PROGRAM DIRECTIVES
C
200    WRITE(IWR,210)
210    FORMAT(
     *2X,'PROGRAM FSM DIRECTIVES:',/,
     *2X,'LD    - LIST PROGRAM DIRECTIVES',/,
     *2X,'SF    - SET/LIST EXTERNAL FILES TO INPUT/OUTPUT',/,
     *2X,'IN    - INITIALIZE PROCESSING PARAMETERS TO DEFAULT ',
     *    'VALUES',/,
     *2X,'SP    - SET/LIST PROCESSING PARAMETERS',/,
     *2X,'FRNT - GENERATE FRONT CURVE (FRONT)',/,
     *2X,'LIST - LIST X,Y COORDINATES OF FRONT CURVE',/,
     *2X,'HELM - APPLY HELMHOLTZ EQUATION SOLVER (MASK,HELM)',/,
     *2X,'RES  - RESUME HELMHOLTZ EQUATION SOLVER (HELM)',/,
     *2X,'TS    - GENERATE TEMPERATURE, SALINITY, DENSITY AND ',
     *    'FREQUENCY PROFILES (TS)',/,
     *2X,'LP    - LIST TEMPERATURE, SALNITY, DENSITY AND FREQUENCY ',
     *    'PROFILES',/,
     *2X,'SIG  - GENERATE SIGMA FIELD (NUVD,NUINT)',/,
     *2X,'SV    - GENERATE SOUND VELOCITY, BLENDED TEMPERATURE OR ',
     *    'BLENDED SALINITY',/,
     *2X,'         OUTPUT (BLENDTS)',/,
     *2X,'END  - END THE PROGRAM')
       GO TO 100
C
C SF - SET UP EXTERNAL FILES TO PROCESS
C
300    CALL SETFS(IRD,IWR,NWFN,JBUF,IH,FA,IA,IST)
       IF(IST .EQ. -1) GO TO 9000
       WRITE(LUCF,REC=2) (JBUF(I),I=1,NWPRCF)
       GO TO 100
C
C IN - INITIALIZE PARAMETERS
C
400    CALL INPAR(MXFCPS,IBUF)
       WRITE(LUCF,REC=1) (IBUF(I),I=1,NWPRCF)
       GO TO 100
C
C SP - SET PARAMETERS
C
500    CALL SETPAR(IRD,IWR,MXFCPS,IBUF,IH,FA,IA,IST)
       IF(IST .EQ. -1) GO TO 9000
       WRITE(LUCF,REC=1) (IBUF(I),I=1,NWPRCF)
       GO TO 100
C
C FRNT - GENERATE FRONT CURVE
C
600    NPTS=IBUF(11)
       LM=IBUF(12)
       RLP=XBUF(13)
       ITER=IBUF(14)
       DLM=LM*2.+1.
C
C OBTAIN CURRENT TIME IN SECONDS AS INITIAL SEED TO CALL
C        RANDOM NUMBER GENERATOR
C
       CALL DATIME(IH,FA,ISEC)
       IF(MOD(ISEC,2) .EQ. 0) ISEC=ISEC+1
       ISEC=99997
C
C CREATE WAVE NUMBER (AK), AMPLITUDE (A), PHASE (PHI)
C
       DO 602 I=0,100
          AK(I)=.25*(.5*I)**2+1.+.3*((RAN(ISEC))-.5)
          A(I)=1./(1.+AK(I)**RLP)
          PHI(I)=2.*3.14159*RAN(ISEC)
602    CONTINUE
C
C CREATE INITIAL CURVE VALUES, RESULT IS A STRAIGHT LINE
```

```
C
      D=NPTS
      D10=D/10.
      DO 604 I=1,NPTS
         D=I
         X(I)=D/D10
         T(I)=X(I)
         Y(I)=(.5+1.5*T(I)/10.)*A(0)*SIN(AK(0)*T(I)+PHI(0))
604   CONTINUE
C
C
      DO 690 K=1,ITER
         WRITE(IWR,610) K
610      FORMAT(2X,'ITERATION = ',I4)
C
C FIRST PASS THROUGH FULL ARRAY GENERATION
C
         YY=A(K)*SIN(AK(K)*T(1)+PHI(K))
         XD=X(3)-X(1)
         YD=Y(3)-Y(1)
         Z=SQRT(XD*XD+YD*YD)
         X1(1)=X(1)-YD*YY/Z
         Y1(1)=Y(1)+XD*YY/Z
         DO 616 I=2,NPTS
            YY=(.5+1.5*(T(I)/T(NPTS)))*A(K)*SIN(AK(K)*T(I)+PHI(K))
C
C LEAST SQUARES FIT OVER +/- LM REPLACES NEED FOR DERIVATIVES
C
            IF(I .LT. NPTS) THEN
C
C NEAR BOUNDARIES CHECK
C
               IF(I .LT. 11 .OR. I .GT. (NPTS-11)) THEN
                  XD=X(I+1)-X(I-1)
                  YD=Y(I+1)-Y(I-1)
               ELSE
                  XD=0.
                  YD=0.
                  DO 614 J=1,10
                     D=J
                     XD=XD+D*(X(I+J)-X(I-J))
                     YD=YD+D*(Y(I+J)-Y(I-J))
614               CONTINUE
               ENDIF
            ENDIF
C
C CREATE ALONGSTREAM VALUE FROM X,Y DELTAS
C
            Z=SQRT(XD*XD+YD*YD)
            X1(I)=X(I)-YD*YY/Z
            Y1(I)=Y(I)+XD*YY/Z
616      CONTINUE
C
C ONCE ROUGH CURVE IS CREATED, GO BACK WITH A SMOOTHING FUNCTION
C
         DO 618 I=0,LM
            X1(-I)=X1(1)
            Y1(-I)=Y1(1)
            X1(NPTS+I)=X1(NPTS)
            Y1(NPTS+I)=Y1(NPTS)
618      CONTINUE
C
C GET FILTERED X,Y VALUES IN "CENTER" OF FILTER WIDTH
C
         DO 630 I=1,NPTS
            X(I)=0.
            Y(I)=0.
            DO 620 J=-LM,LM
               X(I)=X(I)+X1(I+J)
```

```
               Y(I)=Y(I)+Y1(I+J)
620         CONTINUE
            X(I)=X(I)/DLM
            Y(I)=Y(I)/DLM
630      CONTINUE
690   CONTINUE
      IBUF(20)=NPTS
      IF(IBUF(3) .LE. 0) IBUF(3)=IBUF(2)+1
      IREC=IBUF(3)-1
      DO 694 I=1,MXFCPS,NWPRCF/2
         J=I+NWPRCF/2-1
         IREC=IREC+1

         WRITE(LUCF,REC=IREC) (X(K),Y(K),K=I,J)
694   CONTINUE
      IBUF(2)=IREC
      WRITE(LUCF,REC=1) (IBUF(I),I=1,NWPRCF)
      GO TO 100
C
C LIST - LIST X,Y OF FRONT CURVE
C
700   IBP=IA(1) .
      IEP=IA(2)
C
C CHECK NUMBER OF POINTS OF THE FRONT CURVE STORED
C
      IREC=IBUF(3)-1
      NPTS=IBUF(20)
      IF(NPTS .LE. 0) THEN
         WRITE(IWR,710)
710      FORMAT(2X,'FRONT CURVE HAS NOT BEEN GENERATED YET')
         GO TO 100
      ENDIF
C
C
      WRITE(IWR,712) NPTS
712   FORMAT(2X,'TOTAL NUMBER OF FRONT CURVE POINTS: ',I4)
      IF(IBP .EQ. 0) THEN
         WRITE(IWR,714)
714      FORMAT(2X,'ENTER BEGINNING AND ENDING POINTS TO LIST')
         CALL RUI(IRD,IWR,IH,FA,IA,NVS)
      ENDIF
      IBP=MAX0(IA(1),1)
      IEP=MIN0(IA(2),NPTS)
      IEP=MAX0(IBP,IEP)
      WRITE(IWR,716)
716   FORMAT(2X,'POINT',11X,'X',11X,'Y')
      DO 730 I=1,IEP,NWPRCF/2
         J=I+NWPRCF/2-1
         IREC=IREC+1
         READ(LUCF,REC=IREC) (X(K),Y(K),K=I,J)
         DO 720 K=I,J
            IF(K.GE.IBP .AND. K.LE.IEP) THEN
               WRITE(IWR,718) K,X(K),Y(K)
718            FORMAT(2X,I5,2X,F10.5,2X,F10.5)
            ENDIF
720      CONTINUE
730   CONTINUE
      GO TO 100
C
C HELM - APPLY HELMHOLTZ EQUATION SOLVER
C
800   IREC=IBUF(3)-1
      NPTS=IBUF(20)
      IF(NPTS .LE. 0) THEN
         WRITE(IWR,710)
         GO TO 100
      ENDIF
C
```

```
C CHECK IF HMF FILE IS SET
C
      IF(JBUF(21) .EQ. BLANKS) THEN
         WRITE(IWR,802) FILTYP(2)
802      FORMAT(2X,A4,' FILE NAME HAS NOT BEEN DEFINED',/,
     *          2X,'USE DIRECTIVE SF TO SET THE FILE NAME')
         GO TO 100
      ELSE
         WRITE(HMFN,30) (JBUF(I),I=21,20+NWFN)
         CALL CKFILE(IWR,2,HMFN,2,LUS,LUSO,FILTYP,MXROWS,IST)
         IF(IST .EQ. -1) GO TO 100
      ENDIF
C
C READ FRONT CURVE IN
C
812   DO 816 I=1,NPTS,NWPRCF/2
         J=I+NWPRCF/2-1
         IREC=IREC+1
         READ(LUCF,REC=IREC) (X(K),Y(K),K=I,J)
816   CONTINUE
C
C FIND MIN AND MAX X VALUES BETWEEN POINTS 51 AND NPTS-50
C
      XMIN=X(51)
      XMAX=XMIN
      DO 818 I=52,NPTS-50
         IF(X(I) .LT. XMIN) THEN
            XMIN=X(I)
         ELSE
            IF(X(I) .GT. XMAX) XMAX=X(I)
         ENDIF
818   CONTINUE
C
C RESCALE X,Y TO FIT IN MASK BUFFER
C
      D=MXCOLS
      SCALE=D/(XMAX-XMIN)
      MPTS=0
      IX(MPTS)=0
      IY(MPTS)=MXROWS/2
      DO 820 I=51,NPTS-50
         IXX=(X(I)-XMIN)*SCALE+.5
         IYY=Y(I)*SCALE+.5
         IYY=IYY+IY(0)
         IF((IXX.NE.IX(MPTS) .OR. IYY.NE.IY(MPTS)) .AND.
     *      IXX.GE.1) THEN
            MPTS=MPTS+1
            IX(MPTS)=IXX
            IY(MPTS)=IYY
            IF(IX(MPTS) .GE. MXCOLS) GO TO 822
         ENDIF
820   CONTINUE
C
C INITIALIZE MASK BUFFER
C
822   WRITE(IWR,824)
824   FORMAT(2X,'GENERATING Q MASK')
      DO 830 I=1,MXROWS
         DO 828 J=1,MXCOLS
            MASK(J,I)=1
828      CONTINUE
830   CONTINUE
C
C SET LOCATIONS OF FRONT CURVE TO 0'S
C
      DO 832 I=1,MPTS
         MASK(IX(I),IY(I))=0
832   CONTINUE
C
```

```
C SET PIXELS BELOW THE FRONT CURVE TO -1'S
C
      DO 840 I=1,MPTS
         IDIR=0
         JDIR=IY(I)-IY(I-1)
         IF(I .LT. MPTS) THEN
            IF(JDIR .GE. 0) THEN
               IDIR=IX(I)-IX(I-1)
            ELSE
               IDIR=IX(I+1)-IX(I)
            ENDIF
         ELSE
            IDIR=1
         ENDIF
         IF(IDIR .GT. 0) THEN
            DO 836 J=1,IY(I)-1
               K=IY(I)-J
               IF(MASK(IX(I),K) .EQ. 0) GO TO 840
               MASK(IX(I),K)=-1
836         CONTINUE
         ENDIF
840   CONTINUE
C
C PERFORM FIRST ORDER CHECK ON Q BUFFER
C
      NOK=0
      DO 860 I=1,MXCOLS
         IMINUS=0
         IPLUS=0
         DO 850 J=1,MXROWS
            IF(MASK(I,J) .LT. 0) THEN
               IMINUS=IMINUS+1
            ELSE
               IF(MASK(I,J) .GT. 0) IPLUS=IPLUS+1
            ENDIF
850      CONTINUE
         NTOTAL=IMINUS+IPLUS
         IF(NTOTAL.LE.(MXROWS-1) .AND. IMINUS.GT.20 .AND.
     *      IPLUS.GT.20) NOK=NOK+1
860   CONTINUE
      IF(NOK .LT. MXCOLS) THEN
         I=MXCOLS-NOK
         WRITE(IWR,862) I
862      FORMAT(2X,'WARNING - ',I4,' COLUMNS IN Q MASK FAILED ',
     *          'TO PASS FIRST ORDER CHECK')
         WRITE(IWR,864)
864      FORMAT(2X,'CONTINUE, Y/N?')
         CALL RUI(IRD,IWR,IH,FA,IA,NVS)
         IF(IH(1).NE.YES(1) .AND. IH(1).NE.YES(2)) THEN
            CALL CLSFIL(LUSO,MXFS)
            GO TO 100
         ENDIF
      ELSE
         WRITE(IWR,866)
866      FORMAT(2X,'Q MASK PASSED FIRST ORDER CHECK')
      ENDIF
C
C SAVE Q MASK TO CONTROL FILE
C
      IF(IBUF(5) .LE. 0) IBUF(5)=IBUF(2)+1
      IREC=IBUF(5)-1
      DO 880 I=1,MXROWS
         DO 870 J=1,MXCOLS,NWPRCF*2
            K=J+NWPRCF*2-1
            K=MIN0(K,MXCOLS)
            IREC=IREC+1
            WRITE(LUCF,REC=IREC) (MASK(L,I),L=J,K)
870      CONTINUE
880   CONTINUE
```

```
      IBUF(2)=IREC
C
C FIND WHERE FRONT CROSSES LEFT AND RIGHT BOUNDARIES
C
      DO 882 I=1,MXROWS
         IF(MASK(1,I) .EQ. 0) IMINUS=I
         IF(MASK(MXCOLS,I) .EQ. 0) IPLUS=I
882   CONTINUE
C
C SAVE HELM PROCESSING PARAMETERS
C
      IBUF(28)=1
      IBUF(29)=IMINUS
      IBUF(30)=IPLUS
C
C INITIALIZE Q MATRIX BUFFER
C
      DO 892 I=1,MXROWS
         DO 890 J=1,MXCOLS
            Q(J,I)=0.
890      CONTINUE
892   CONTINUE
      GO TO 930
C
C RES - RESUME HELM PROCESS
C
900   IF(IBUF(5) .LE. 0) THEN
         WRITE(IWR,902)
902      FORMAT(2X,'Q MASK HAS NOT BEEN GENERATED YET - EXECUTE ',
     *          'DIRECTIVE HELM FIRST')
      ENDIF
C
C CHECK EXISTENCE OF HMF FILE
C
      IF(JBUF(21) .EQ. BLANKS) THEN
         WRITE(IWR,802) FILTYP(2)
         GO TO 100
      ELSE
         WRITE(HMFN,30) (JBUF(I),I=21,20+NWFN)
         CALL CKFILE(IWR,1,HMFN,2,LUS,LUSO,FILTYP,MXROWS,IST)
         IF(IST .NE. 0) GO TO 100
      ENDIF
C
C READ IN Q MATRIX
C
      WRITE(IWR,910) FILTYP(2)
910   FORMAT(2X,'READING ',A4,' FILE')
      DO 912 I=1,MXCOLS
         READ(LUHMF,REC=I) (Q(I,J),J=1,MXROWS)
912   CONTINUE
C
C READ IN Q MASK FROM CONTROL FILE
C
      IREC=IBUF(5)-1
      DO 920 I=1,MXROWS
         DO 914 J=1,MXCOLS,NWPRCF*2
            K=J+NWPRCF*2-1
            K=MIN0(K,MXCOLS)
            IREC=IREC+1
            READ(LUCF,REC=IREC) (MASK(L,I),L=J,K)
914      CONTINUE
920   CONTINUE
C
C
930   ALFA=XBUF(21)
      BETA=XBUF(22)
      H=XBUF(23)
      MAXH=IBUF(24)
      MINH=IBUF(25)
```

```
      JTER=IBUF(26)
      ISTART=IBUF(28)
      IMINUS=IBUF(29)
      IPLUS=IBUF(30)
      WRITE(IWR,932) ISTART,JTER
932   FORMAT(
     *2X,'EXECUTING HELMHOLTZ EQUATION SOLVER',/,
     *2X,'STARTING ITERATION NUMBER: ',I4,', LAST ITERATION ',
     *   'NUMBER: ',I4)
      IF(ISTART .GT. JTER) THEN
         WRITE(IWR,934)
934      FORMAT(2X,'HELM PROCESS WAS COMPLETED')
         CALL CLSFIL(LUSO,MXFS)
         GO TO 100
      ENDIF
      B2=1./(BETA*BETA)
      H2=H*H
      SLAM2=B2*H2
      SLAM=SQRT(SLAM2)
      ETA(-1)=MINH
      ETA(0)=0.
      ETA(1)=MAXH
C
C CHECK IF BOUNDARIES SHOULD BE INITIALIZED
C
      IF(ISTART .EQ. 1) THEN
         DO 936 I=1,MXCOLS
            Q(I,1)=MINH
            Q(I,MXROWS)=MAXH
936      CONTINUE
         DO 938 I=1,MXROWS
            J=MASK(1,I)
            D=IABS(I-IMINUS)
            Q(1,I)=ETA(J)*(1.-EXP(-SLAM*D))
            J=MASK(MXROWS,I)
            D=IABS(I-IPLUS)
            Q(MXCOLS,I)=ETA(J)*(1.-EXP(-SLAM*D))
938      CONTINUE
      ENDIF
C
C HELM SOLVER
C
      DO 970 I=ISTART,JTER
         RMAX=0.
         DO 960 J=2,MXCOLS-1
            DO 950 K=2,MXROWS-1
               L=MASK(J,K)
               IF(L .NE. 0) THEN
                  R=ALFA*(.25*(Q(J-1,K)+Q(J+1,K)+Q(J,K+1)+
     *               Q(J,K-1)-4.*Q(J,K))-SLAM2*(Q(J,K)-ETA(L)))
               ELSE
                  R=0.
               ENDIF
               Q(J,K)=Q(J,K)+R
               R=ABS(R)
               IF(RMAX .LT. R) RMAX=R
950         CONTINUE
960      CONTINUE
         WRITE(IWR,962) I,RMAX
962      FORMAT(2X,'ITERATION: ',I4,', MAXIMUM ERROR: ',G20.8)
         IF(MOD(I,50) .EQ. 1) THEN
            DO 964 K=1,MXCOLS
               WRITE(LUHMF,REC=K) (Q(K,J),J=1,MXROWS)
964         CONTINUE
            IBUF(28)=I+1
            WRITE(LUCF,REC=1) (IBUF(K),K=1,NWPRCF)
            CALL DATIME(IH,IA,ISEC)
            JBUF(NWFN+21)=IH(1)
            JBUF(NWFN+22)=IH(2)
```

```
            JBUF(NWFN+23)=IA(1)
            JBUF(NWFN+24)=IA(2)
            WRITE(LUCF,REC=2) (JBUF(K),K=1,NWPRCF)
         ENDIF
970   CONTINUE
      IF(MOD(JTER,50) .NE. 1) THEN
         DO 974 K=1,MXCOLS
            WRITE(LUHMF,REC=K) (Q(K,J),J=1,MXROWS)
974      CONTINUE
         IBUF(28)=JTER+1
         WRITE(LUCF,REC=1) (IBUF(K),K=1,NWPRCF)
         CALL DATIME(IH,IA,ISEC)
         JBUF(NWFN+21)=IH(1)
         JBUF(NWFN+22)=IH(2)
         JBUF(NWFN+23)=IA(1)
         JBUF(NWFN+24)=IA(2)
         WRITE(LUCF,REC=2) (JBUF(K),K=1,NWPRCF)
      ENDIF
      CALL CLSFIL(LUSO,MXFS)
      GO TO 100
C
C TS - CREATE TEMPERATURE AND SALINITY PROFILES
C
1000  MAXH=IBUF(24)
      MINH=IBUF(25)
      LAT=IBUF(31)
      LONG=IBUF(32)
      IZ=IBUF(33)
      I=ZLEV(MXZS)
      NZS=I/IZ+1
      IF(MOD(I,IZ) .NE. 0) THEN
         WRITE(IWR,1002) I,IZ
1002     FORMAT(2X,'MAXIMUM DEPTH ',I4,' IS NOT EVENLY DIVISIBLE ',
     *          'BY PARAMETER IZ: ',I4)
         GO TO 100
      ENDIF
      I=IABS(MAXH)/IZ+1
      J=IABS(MINH)/IZ+1
      MZS=NZS+I+J+1
      IF(MZS .GT. MXVS) THEN
         WRITE(IWR,1004)
1004     FORMAT(2X,'PARAMETER IZ IS TOO SMALL')
         GO TO 100
      ENDIF
C
C CHECK EXISTENCE OF LEVITUS DATABASE FILE
C
      IF(JBUF(1) .EQ. BLANKS) THEN
         WRITE(IWR,802) FILTYP(3)
      ELSE
         WRITE(DFN,80) (JBUF(I),I=1,12)
         CALL CKFILE(IWR,1,DFN,3,LUS,LUSO,FILTYP,180,IST)
         IF(IST .NE. 0) GO TO 100
      ENDIF
C
C READ TEMPERATURE AND SALINITY PROFILES IN FROM LEVITUS DATABASE
C
      CALL RDTS(LUDBF,LONG,LAT,TEMP,SAL,180,XDBF)
      CALL CLSFIL(LUSO,MXFS)
C
C SET UP DEPTH INDEX TABLE
C
      DO 1024 I=1,NZS
         DEPO(I)=(I-1)*IZ
1024  CONTINUE
C
C INTERPOLATE TEMPERATURE AND SALINITY PROFILES
C
      CALL INTRPL(IWR,MXZS,ZLEV,TEMP,NZS,DEPO,TEMPO)
```

```
      CALL INTRPL(IWR,MXZS,ZLEV,SAL,NZS,DEPO,SALO)
C
C
      PR=0.
      DO 1030 I=1,NZS-1
C
C COMPUTE BRUNT-VAISALA FREQUENCIES
C
         BVFO(I)=BVFRQ(SALO(I),TEMPO(I),DEPO(I),2,PAV,E)
C
C USE SIMPLE TEMPERATURE INSTEAD OF POTENTIAL TEMPERATURE (THETA)
C
         DUM=SVAN(SALO(I),TEMPO(I),PR,SIGO(I))
1030  CONTINUE
      BVFO(NZS)=BVFO(NZS-1)+(BVFO(NZS-1)-BVFO(NZS-2))
      DUM=SVAN(SALO(NZS),TEMPO(NZS),PR,SIGO(NZS))
C
C FIND SIGMA VALUE BASED UPON MAXIMUM BVF VALUE
C
      IDXMAX=1
      BVFMAX=BVFO(IDXMAX)
      DO 1040 I=2,NZS
         IF(BVFO(I) .GT. BVFMAX) THEN
            IDXMAX=I
            BVFMAX=BVFO(I)
         ENDIF
1040  CONTINUE
      S=SIGO(IDXMAX)
      WRITE(IWR,1042) S,BVFMAX
1042  FORMAT(2X,'SIGMA VALUE: ',F10.5,' FOUND AT BVF VALUE: ',F10.5)
C
C SAVE MAXIMUM SIGMA VALUE TO CONTROL FILE
C
      IBUF(35)=MAXH
      IBUF(36)=MINH
      IBUF(37)=IZ
      IBUF(38)=NZS
      IBUF(39)=IDXMAX
      XBUF(40)=S
C
C STORE DEPTH, TEMPERATURE, SALINITY, SIGMA, BVF PROFILES TO
C       CONTROL FILE
C
      IF(IBUF(7) .LE. 0) IBUF(7)=IBUF(2)+1
      IREC=IBUF(7)-1
      DO 1050 I=1,MXVS,NWPRCF/5
         J=I+NWPRCF/5-1
         IREC=IREC+1
         WRITE(LUCF,REC=IREC) (DEPO(K),TEMPO(K),SALO(K),SIGO(K),
     *                         BVFO(K),K=I,J)
1050  CONTINUE
      IBUF(2)=IREC
      WRITE(LUCF,REC=1) (IBUF(I),I=1,NWPRCF)
      GO TO 100
C
C LP - LIST DENSITY AND FREQUENCY PROFILES
C
1100  IBP=IA(1)
      IEP=IA(2)
      IF(IBUF(7) .LE. 0) THEN
         WRITE(IWR,1102)
1102     FORMAT(2X,'NO DENSITY AND BVF DATA STORED IN THE CONTROL ',
     *          'FILE - EXECUTE DIRECTIVE TS FIRST')
         GO TO 100
      ENDIF
C
C
      IREC=IBUF(7)-1
      NZS=IBUF(38)
```

```
      WRITE(IWR,1104) NZS
1104  FORMAT(2X,'TOTAL NUMBER OF DENSITY AND FREQUENCY VALUES ',
     *           'STORED: ',I4)
      IF(IBP .EQ. 0) THEN
         WRITE(IWR,1106)
1106     FORMAT(2X,'ENTER BEGINNING AND ENDING VALUES TO LIST')
         CALL RUI(IRD,IWR,IH,FA,IA,NVS)
      ENDIF
      IBP=MAX0(IA(1),1)
      IEP=MIN0(IA(2),NZS)
      IEP=MAX0(IBP,IEP)
      WRITE(IWR,1110)
1110  FORMAT(2X,'VALUE',8X,'DEPTH',2X,'TEMPERATURE',5X,'SALINITY',
     *       6X,'DENSITY',4X,'FREQUENCY')
      DO 1130 I=1,IEP,NWPRCF/5
         J=I+NWPRCF/5-1
         IREC=IREC+1
         READ(LUCF,REC=IREC) (DEP(K),TEMP(K),SAL(K),SIG(K),
     *                       BVF(K),K=I,J)
         DO 1128 K=I,J
            IF(K.GE.IBP .AND. K.LE.IEP) THEN
               WRITE(IWR,1126) K,DEP(K),TEMP(K),SAL(K),SIG(K),
     *                         BVF(K)
1126           FORMAT(2X,I5,5(3X,F10.5))
            ENDIF
1128     CONTINUE
1130  CONTINUE
      GO TO 100
C
C SIG - GENERATE SIGMA FIELD
C
1200  IF(IBUF(7) .LE. 0) THEN
         WRITE(IWR,1102)
         GO TO 100
      ENDIF
      MAXH=IBUF(35)
      MINH=IBUF(36)
      IZ=IBUF(37)
      NZS=IBUF(38)
C
C CHECK EXISTENCE OF Q MATRIX FILE
C
      IF(JBUF(21) .EQ. BLNAKS) THEN
         WRITE(IWR,802) FILTYP(2)
         GO TO 100
      ELSE
         WRITE(HMFN,30) (JBUF(I),I=21,20+NWFN)
         CALL CKFILE(IWR,1,HMFN,2,LUS,LUSO,FILTYP,MXROWS,IST)
         IF(IST .NE. 0) GO TO 100
      ENDIF
C
C CHECK SFF FILE NAME
C
      IF(JBUF(31) .EQ. BLANKS) THEN
         WRITE(IWR,802) FILTYP(4)
         CALL CLSFIL(LUSO,MXFS)
         GO TO 100
      ELSE
         WRITE(SFFN,30) (JBUF(I),I=31,30+NWFN)
         CALL CKFILE(IWR,2,SFFN,4,LUS,LUSO,FILTYP,NZS,IST)
         IF(IST .EQ. -1) THEN
            CALL CLSFIL(LUSO,MXFS)
            GO TO 100
         ENDIF
      ENDIF
C
C GET FIRST END POINT
C
      WRITE(IWR,1220)
```

```
1220  FORMAT(2X,'ENTER X,Y OF FIRST END POINT')
      CALL RUI(IRD,IWR,IH,FA,IA,NVS)
      IF(IA(1).LT.1 .OR. IA(1).GT.MXCOLS) THEN
         WRITE(IWR,1222) MXCOLS
1222     FORMAT(2X,'X COORDINATE CANNOT BE < 1 OR > ',I4)
         CALL CLSFIL(LUSO,MXFS)
         GO TO 100
      ELSE
         IF(IA(2).LT.1 .OR. IA(2).GT.MXROWS) THEN
            WRITE(IWR,1224) MXROWS
1224        FORMAT(2X,'Y COORDINATE CANNOT BE < 1 OR > ',I4)
            CALL CLSFIL(LUSO,MXFS)
            GO TO 100
         ENDIF
      ENDIF
      IX1=IA(1)
      IY1=IA(2)
C
C GET SECOND END POINT
C
      WRITE(IWR,1226)
1226  FORMAT(2X,'ENTER X,Y OF SECOND END POINT')
      CALL RUI(IRD,IWR,IH,FA,IA,NVS)
      IF(IA(1).LT.1 .OR. IA(1).GT.MXCOLS) THEN
         WRITE(IWR,1222) MXCOLS
         CALL CLSFIL(LUSO,MXFS)
         GO TO 100
      ELSE
         IF(IA(2).LT.1 .OR. IA(2).GT.MXROWS) THEN
            WRITE(IWR,1224) MXROWS
            CALL CLSFIL(LUSO,MXFS)
            GO TO 100
         ENDIF
      ENDIF
      IX2=IA(1)
      IY2=IA(2)
C
C INTERPOLATE THE POINTS BETWEEN TWO END POINTS
C
      CALL DLXY(IX1,IY1,IX2,IY2,MPTS,MXVS,IXS,IYS)
      IF(MPTS .GT. MXVS) THEN
         WRITE(IWR,1228) MXVS
1228     FORMAT(2X,'MORE THAN ',I4,' POINTS ARE FOUND BETWEEN ',
     *             'THE TWO END POINTS')
         GO TO 100
      ENDIF
C
C READ IN TEMPERATURE, SALINITY, SIGMA AND BVF PROFILES FROM THE CONTROL FILE
C
      IREC=IBUF(7)-1
      DO 1230 I=1,NZS,NWPRCF/5
         J=I+NWPRCF/5-1
         IREC=IREC+1
         READ(LUCF,REC=IREC) (DEP(K),TEMP(K),SAL(K),SIG(K),
     *                        BVF(K),K=I,J)
1230  CONTINUE
C
C SET UP DEPTH RANGE INCLUDING MINH AND MAXH AND SHIFT SIGMA AND BVF
C     DATA ACCORDINGLY
C
      MZS=IABS(MINH/IZ)+1
      DO 1232 I=NZS,1,-1
         DEP(MZS+I)=(I-1)*IZ
         BVF(MZS+I)=BVF(I)
         SIG(MZS+I)=SIG(I)
1232  CONTINUE
      DO 1234 I=MZS,1,-1
         DEP(I)=DEP(I+1)-IZ
         BVF(I)=BVF(I+1)
```

```
         SIG(I)=SIG(I+1)
1234  CONTINUE
      J=MZS+NZS
      MZS=IABS(MAXH)/IZ+1
      DO 1236 I=1,MZS
         DEP(J+I)=DEP(J+I-1)+IZ
         BVF(J+I)=BVF(J+I-1)
         SIG(J+I)=SIG(J+I-1)
1236  CONTINUE
      MZS=J+MZS
C
C FIND DEPTH FOR THE MAXIMUM BVF VALUE
C
      BVFMAX=BVF(1)
      DEPTH=DEP(1)
      DO 1238 I=2,MZS
         IF(BVF(I) .GT. BVFMAX) THEN
            BVFMAX=BVF(I)
            DEPTH=DEP(I)
         ENDIF
1238  CONTINUE
C
C
      IREC=0
      DO 1260 I=1,MPTS
         IF(IREC .NE. IXS(I)) THEN
            IREC=IXS(I)
            READ(LUHMF,REC=IREC) (Q(IREC,J),J=1,MXROWS)
         ENDIF
         QV=Q(IREC,IYS(I))
         IF(QV .LE. DEPTH) THEN
            LZS=0
         ELSE
            D=QV-DEPTH
            LZS=D/IZ
            D1=LZS*IZ
            IF(D1 .LT. D) LZS=LZS+1
         ENDIF
         K=0
         DO 1242 J=LZS,1,-1
            K=K+1
            DEPO(J)=QV-(K*IZ)
1242     CONTINUE
         DO 1244 J=1,NZS
            DEPO(LZS+J)=QV+(J-1)*IZ
1244     CONTINUE
         KZS=NZS+LZS
         IDX=LZS+1
         CALL INTRPL(IWR,MZS,DEP,BVF,KZS,DEPO,BVFO)
         CALL INTRPL(IWR,MZS,DEP,SIG,KZS,DEPO,SIGO)
         CALL SIGINT(IZ,KZS,BVFO,SIGO)
         WRITE(LUSFF,REC=I) (SIGO(J),J=IDX,KZS)
         IF(I .EQ. 1) THEN
            SIGMIN=SIGO(IDX)
            IA(1)=I
            IA(2)=1
            SIGMAX=SIGMIN
            IA(3)=IA(1)
            IA(4)=IA(2)
         ENDIF
         DO 1250 J=IDX,KZS
            IF(SIGO(J) .LT. SIGMIN) THEN
               SIGMIN=SIGO(J)
               IA(1)=I
               IA(2)=J-IDX+1
            ELSE
               IF(SIGO(J) .GT. SIGMAX) THEN
                  SIGMAX=SIGO(J)
                  IA(3)=I
```

```
                IA(4)=J-IDX+1
             ENDIF
          ENDIF
1250    CONTINUE
        IF(MOD(I,50) .EQ. 0) THEN
           WRITE(IWR,1252) I,MPTS
1252       FORMAT(2X,'COMPLETED POINT NUMBER: ',I4,', TOTAL NUMBER ',
     *                 'OF POINTS TO PROCESS: ',I4)
        ENDIF
1260  CONTINUE
      IF(MOD(MPTS,50) .NE. 0) WRITE(IWR,1252) MPTS,MPTS
      CALL CLSFIL(LUSO,MXFS)
      IBUF(41)=IX1
      IBUF(42)=IY1
      IBUF(43)=IX2
      IBUF(44)=IY2
      XBUF(45)=SIGMIN
      XBUF(46)=SIGMAX
      IBUF(47)=MINH
      IBUF(48)=MAXH
      IBUF(49)=IZ
      IBUF(50)=NZS
      WRITE(IWR,1262) SIGMIN,IA(1),IA(2)
1262  FORMAT(2X,'MINIMUM SIGMA VALUE: ',F10.5,' FOUND AT X,Y ',
     *          'LOCATION: ',I4,',',I4)
      WRITE(IWR,1264) SIGMAX,IA(3),IA(4)
1264  FORMAT(2X,'MAXIMUM SIGMA VALUE: ',F10.5,' FOUND AT X,Y ',
     *          'LOCATION: ',I4,',',I4)
      WRITE(LUCF,REC=1) (IBUF(I),I=1,NWPRCF)
      CALL DATIME(IH,IA,ISEC)
      JBUF(31+NWFN)=IH(1)
      JBUF(32+NWFN)=IH(2)
      JBUF(33+NWFN)=IA(1)
      JBUF(34+NWFN)=IA(2)
      WRITE(LUCF,REC=2) (JBUF(I),I=1,NWPRCF)
      GO TO 100
C
C SV - GENERATE SOUND VELOCITY, TEMPERATURE OR SALINITY OUTPUT
C
1300  IF(JBUF(1) .EQ. BLANKS) THEN
         WRITE(IWR,802) FILTYP(3)
      ELSE
         WRITE(DFN,80) (JBUF(I),I=1,12)
         CALL CKFILE(IWR,1,DFN,3,LUS,LUSO,FILTYP,180,IST)
         IF(IST .NE. 0) GO TO 100
      ENDIF
C
C CHECK EXISTENCE OF HMF FILE
C
      IF(JBUF(21) .EQ. BLANKS) THEN
         WRITE(IWR,802) FILTYP(2)
         CALL CLSFIL(LUSO,MXFS)
         GO TO 100
      ELSE
         WRITE(HMFN,30) (JBUF(I),I=21,20+NWFN)
         CALL CKFILE(IWR,1,HMFN,2,LUS,LUSO,FILTYP,MXROWS,IST)
         IF(IST .NE. 0) THEN
            CALL CLSFIL(LUSO,MXFS)
            GO TO 100
         ENDIF
      ENDIF
C
C
      IX1=IBUF(41)
      IY1=IBUF(42)
      IX2=IBUF(43)
      IY2=IBUF(44)
      SIGMIN=XBUF(45)
      SIGMAX=XBUF(46)
```

```
      MINH=IBUF(47)
      MAXH=IBUF(48)
      IZ=IBUF(49)
      NZS=IBUF(50)
      E=XBUF(51)
      IRCF=IBUF(52)
C
C CHECK EXISTENCE OF SFF FILE
C
      IF(JBUF(31) .EQ. BLANKS) THEN
         WRITE(IWR,802) FILTYP(4)
         CALL CLSFIL(LUSO,MXFS)
         GO TO 100
      ELSE
         WRITE(SFFN,30) (JBUF(I),I=31,30+NWFN)
         CALL CKFILE(IWR,1,SFFN,4,LUS,LUSO,FILTYP,NZS,IST)
         IF(IST .NE. 0) THEN
            CALL CLSFIL(LUSO,MXFS)
            GO TO 100
         ENDIF
      ENDIF
C
C CHECK IF SOUND VELOCITY OUTPUT IS REQUIRED
C
      IF(JBUF(41) .EQ. BLANKS) THEN
         WRITE(IWR,1302) FILTYP(5)
1302     FORMAT(2X,A4,'FILE NAME IS NOT DEFINED - NO SOUND VELOCITY ',
     *          'OUTPUT')
      ELSE
         WRITE(SVFN,30) (JBUF(I),I=41,40+NWFN)
         CALL CKFILE(IWR,2,SVFN,5,LUS,LUSO,FILTYP,NZS,IST)
         IF(IST .EQ. -1) THEN
            CALL CLSFIL(LUSO,MXFS)
            GO TO 100
         ENDIF
      ENDIF
C
C CHECK IF TEMPERATURE OUTPUT IS REQUIRED
C
      IF(JBUF(51) .EQ. BLANKS) THEN
         WRITE(IWR,1304) FILTYP(6)
1304     FORMAT(2X,A4,'FILE NAME IS NOT DEFINED - NO TEMPERATURE ',
     *          'OUTPUT')
      ELSE
         WRITE(TFN,30) (JBUF(I),I=51,50+NWFN)
         CALL CKFILE(IWR,2,TFN,6,LUS,LUSO,FILTYP,NZS,IST)
         IF(IST .EQ. -1) THEN
            CALL CLSFIL(LUSO,MXFS)
            GO TO 100
         ENDIF
      ENDIF
C
C CHECK IF SALINITY OUTPUT IS REQUIRED
C
      IF(JBUF(61) .EQ. BLANKS) THEN
         WRITE(IWR,1306) FILTYP(7)
1306     FORMAT(2X,A4,'FILE NAME IS NOT DEFINED - NO SALINITY OUTPUT')
      ELSE
         WRITE(SFN,30) (JBUF(I),I=61,60+NWFN)
         CALL CKFILE(IWR,2,SFN,7,LUS,LUSO,FILTYP,NZS,IST)
         IF(IST .EQ. -1) THEN
            CALL CLSFIL(LUSO,MXS)
            GO TO 100
         ENDIF
      ENDIF
C
C AT LEAST ONE OUTPUT IS REQUESTED
C
      IF(LUSVF.LE.0 .AND. LUTF.LE.0 .AND. LUSF.LE.0) THEN
```

```
      WRITE(IWR,1308)
1308      FORMAT(2X,'NONE OF THE OUTPUT FILE NAMES ARE DEFINED',/,
     *         2X,'USE DIRECTIVE SF TO SET AT LEAST ONE OUTPUT ',
     *            'FILE NAME')
         CALL CLSFIL(LUSO,MXFS)
         GO TO 100
      ENDIF
      WRITE(IWR,1310) IX1,IY1,IX2,IY2
1310  FORMAT(2X,'TWO END POINTS PREVIOUSLY DEFINED: (',I4,',',I4,
     *          ');',3X,'(',I4,',',I4,')')
C
C USER INPUTS TWO SETS OF LATITUDE AND LONGITUDE
C
      WRITE(IWR,1312) MINH
1312  FORMAT(2X,'ENTER LATITUDE AND LONGITUDE CORRESPONDING TO ',
     *          'MINIMUM H VALUE: ',I4)
      CALL RUI(IRD,IWR,IH,FA,IA,NVS)
      IF(NVS .EQ. 0) THEN
         CALL CLSFIL(LUSO,MXFS)
         GO TO 100
      ENDIF
      IF(IA(1).LT.-90 .OR. IA(1).GT.90) THEN
         WRITE(IWR,1314)
1314     FORMAT(2X,'LATITUDE MUST BE BETWEEN -90 AND 90 DEGREES')
         CALL CLSFIL(LUSO,MXFS)
         GO TO 100
      ELSE
         IF(IA(2).LT.-180 .OR. IA(1).GT.180) THEN
            WRITE(IWR,1316)
1316        FORMAT(2X,'LONGITUDE MUST BE BETWEEN -180 AND 180 DEGREES')
            CALL CLSFIL(LUSO,MXFS)
            GO TO 100
         ENDIF
      ENDIF
      LAT=IA(1)
      LONG=IA(2)
C
C
      WRITE(IWR,1318) MAXH
1318  FORMAT(2X,'ENTER LATITUDE AND LONGITUDE CORRESPONDING TO ',
     *          'MAXIMUM H VALUE: ',I4)
      CALL RUI(IRD,IWR,IH,FA,IA,NVS)
      IF(NVS .EQ. 0) THEN
         CALL CLSFIL(LUSO,MXFS)
         GO TO 100
      ENDIF
      IF(IA(1).LT.-90 .OR. IA(1).GT.90) THEN
         WRITE(IWR,1314)
         CALL CLSFIL(LUSO,MXFS)
         GO TO 100
      ELSE
         IF(IA(2).LT.-180 .OR. IA(2).GT.180) THEN
            WRITE(IWR,1316)
            CALL CLSFIL(LUSO,MXFS)
            GO TO 100
         ENDIF
      ENDIF
      LAT2=IA(1)
      LONG2=IA(2)
C
C READ IN TEMPERATURE AND SALINITY PROFILES AT THESE TWO LOCATIONS
C
      CALL RDTS(LUDBF,LONG,LAT,TEMP,SAL,180,XDBF)
      CALL RDTS(LUDBF,LONG2,LAT2,TEMP2,SAL2,180,XDBF)
C
C FIND THE POINTS BETWEEN TWO END POINTS
C
      CALL DLXY(IX1,IY1,IX2,IY2,MPTS,MXVS,IXS,IYS)
C
```

```
C COMPUTE SIGMA PROFILES AT THESE TWO LOCAITONS
C
      DO 1320 I=1,NZS
         DEP(I)=(I-1)*IZ
         DEPO2(I)=DEP(I)
1320  CONTINUE
      CALL INTRPL(IWR,MXZS,ZLEV,TEMP,NZS,DEPO2,TEMPO)
      CALL INTRPL(IWR,MXZS,ZLEV,SAL,NZS,DEPO2,SALO)
      CALL INTRPL(IWR,MXZS,ZLEV,TEMP2,NZS,DEPO2,TEMPO2)
      CALL INTRPL(IWR,MXZS,ZLEV,SAL2,NZS,DEPO2,SALO2)
      PR=0.
      DO 1322 I=1,NZS-1
         BVFO(I)=BVFRQ(SALO(I),TEMPO(I),DEPO2(I),2,PAV,EE)
         BVFO2(I)=BVFRQ(SALO2(I),TEMPO2(I),DEPO2(I),2,PAV,EE)
         DUM=SVAN(SALO(I),TEMPO(I),PR,SIGO(I))
         DUM=SVAN(SALO2(I),TEMPO2(I),PR,SIGO2(I))
1322  CONTINUE
      BVFO(NZS)=BVFO(NZS-1)+(BVFO(NZS-1)-BVFO(NZS-2))
      BVFO2(NZS)=BVFO2(NZS-1)+(BVFO2(NZS-1)-BVFO2(NZS-2))
      DUM=SVAN(SALO(NZS),TEMPO(NZS),PR,SIGO(NZS))
      DUM=SVAN(SALO2(NZS),TEMPO2(NZS),PR,SIGO2(NZS))
      CALL SIGINT(IZ,NZS,BVFO,SIGO)
      CALL SIGINT(IZ,NZS,BVFO2,SIGO2)
C
C
      WRITE(IWR,1330) FILTYP(4),SIGMIN,SIGMAX,LAT,LONG,SIGO(1),
     *                SIGO(NZS),LAT2,LONG2,SIGO2(1),SIGO2(NZS)
1330  FORMAT(
     *2X,'MIN/MAX SIGMA VALUES FOUND IN ',A4,' FILE: ',F10.5,',',
     *   F10.5,/,
     *2X,'MIN/MAX SIGMA VALUES FOUND AT LAT ',I4,', LONG ',I4,': ',
     *   F10.5,',',F10.5,/,
     *2X,'MIN/MAX SIGMA VALUES FOUND AT LAT ',I4,', LONG ',I4,': ',
     *   F10.5,',',F10.5)
      SIGMIN=AMIN1(SIGMIN,SIGO(1),SIGO2(1))
      SIGMAX=AMAX1(SIGMAX,SIGO(NZS),SIGO2(NZS))
C
C EXTRAPOLATE TEMPERATURE, SALINITY, SIGMA DATA
C
      MZS=IABS(MINH)/IZ+1
      DO 1340 I=NZS,1,-1
         TEMPO(MZS+I)=TEMPO(I)
         TEMPO2(MZS+I)=TEMPO2(I)
         SALO(MZS+I)=SALO(I)
         SALO2(MZS+I)=SALO2(I)
         SIGO(MZS+I)=SIGO(I)
         SIGO2(MZS+I)=SIGO2(I)
1340  CONTINUE
      DTO=TEMPO(MZS+2)-TEMPO(MZS+1)
      DTO2=TEMPO2(MZS+2)-TEMPO2(MZS+1)
      DSO=SALO(MZS+2)-SALO(MZS+1)
      DSO2=SALO2(MZS+2)-SALO2(MZS+1)
      DSIGO=(SIGO(MZS+1)-SIGMIN)/MZS
      IF(DSIGO .EQ. 0.) DSIGO=.000005
      DSIGO2=(SIGO2(MZS+1)-SIGMIN)/MZS
      IF(DSIGO2 .EQ. 0.) DSIGO2=.000005
      DO 1342 I=MZS,1,-1
         TEMPO(I)=TEMPO(I+1)-DTO
         TEMPO2(I)=TEMPO2(I+1)-DTO2
         SALO(I)=SALO(I+1)-DSO
         SALO2(I)=SALO2(I+1)-DSO2
         SIGO(I)=SIGO(I+1)-DSIGO
         SIGO2(I)=SIGO2(I+1)-DSIGO2
1342  CONTINUE
      J=MZS+NZS
      MZS=IABS(MAXH)/IZ+1
      DTO=TEMPO(J)-TEMPO(J-1)
      DTO2=TEMPO2(J)-TEMPO2(J-1)
      DSO=SALO(J)-SALO(J-1)
```

```
      DSO2=SALO2(J)-SALO2(J-1)
      DSIGO=(SIGMAX-SIGO(J))/MZS
      IF(DSIGO .EQ. 0.) DSIGO=.000005
      DSIGO2=(SIGMAX-SIGO2(J))/MZS
      IF(DSIGO2 .EQ. 0.) DSIGO2=.000005
      DO 1344 I=1,MZS
         TEMPO(J+I)=TEMPO(J+I-1)+DTO
         TEMPO2(J+I)=TEMPO2(J+I-1)+DTO2
         SALO(J+I)=SALO(J+I-1)+DSO
         SALO2(J+I)=SALO2(J+I-1)+DSO2
         SIGO(J+I)=SIGO(J+I-1)+DSIGO
         SIGO2(J+I)=SIGO2(J+I-1)+DSIGO2
1344  CONTINUE
      MZS=J+MZS
      SIGMIN=AMIN1(SIGMIN,SIGO(1),SIGO2(1))
      SIGMAX=AMAX1(SIGMAX,SIGO(MZS),SIGO2(MZS))
      KZS=MZS*1.5
      IF(KZS .GT. MXVS) THEN
         WRITE(IWR,1346)
1346     FORMAT(2X,'TOO MANY WORDS REQUIRED TO PERFORM INTERPOLATION')
         CALL CLSFIL(LUSO,MXFS)
         GO TO 100
      ENDIF
C
C COMPUTE DELTA SIGMA BASED UPON MINIMUM SIGMA AND MAXIMUM SIGMA VALUES
C
      DELSIG=(SIGMAX-SIGMIN)/(KZS-1)
      DO 1348 I=1,KZS
         SIG2(I)=SIGMIN+(I-1)*DELSIG
1348  CONTINUE
C
C
      CALL INTRPL(IWR,MZS,SIGO,TEMPO,KZS,SIG2,TEMP)
      CALL INTRPL(IWR,MZS,SIGO2,TEMPO2,KZS,SIG2,TEMP2)
      CALL INTRPL(IWR,MZS,SIGO,SALO,KZS,SIG2,SAL)
      CALL INTRPL(IWR,MZS,SIGO2,SALO2,KZS,SIG2,SAL2)
C
C COMPUTE TEMPERATURE AND SALINITY FIELDS BASED UPON SIGMA
C
      PR=0.
      IREC=0
      DO 1390 I=1,MPTS
         READ(LUSFF,REC=I) (SIG(J),J=1,NZS)
         CALL LINTPL(IWR,NZS,SIG,DEP,KZS,SIG2,DEPO,INDEX)
         IF(IREC .NE. IXS(I)) THEN
            IREC=IXS(I)
            READ(LUHMF,REC=IREC) (Q(IREC,J),J=1,MXROWS)
         ENDIF
         QV=Q(IREC,IYS(I))
         QE1=(MAXH-QV)**E
         QE2=(QV-MINH)**E
         QE12=QE1+QE2
         DO 1358 J=1,KZS
            TEMPO(J)=(QE1*TEMP(J)+QE2*TEMP2(J))/QE12
            SALO(J)=(QE1*SAL(J)+QE2*SAL2(J))/QE12
1358     CONTINUE
         CALL INTRPL(IWR,KZS,DEPO,TEMPO,NZS,DEPO2,TEMPO2)
         CALL INTRPL(IWR,KZS,DEPO,SALO,NZS,DEPO2,SALO2)
C
C CHECK IF RECOMPUTING SIGMA FIELD IS NECESSARY
C
         IF(IRCF.EQ.YES(1) .OR. IRCF.EQ.YES(2)) THEN
            DO 1364 J=1,NZS
               DUM=SVAN(SALO2(J),TEMPO2(J),DEPO2(J),SIGO2(J))
1364        CONTINUE
            ISWAP=0
            DO 1370 J=1,NZS-1
               DO 1368 K=J+1,NZS
                  IF(SIGO2(J) .GT. SIGO2(K)) THEN
```

```
                  IF(ISWAP .EQ. 0) ISWAP=J
                  S=SIGO2(J)
                  SIGO2(J)=SIGO2(K)
                  SIGO2(K)=S
                  S=TEMPO2(J)
                  TEMPO2(J)=TEMPO2(K)
                  TEMPO2(K)=S
                  S=SALO2(J)
                  SALO2(J)=SALO2(K)
                  SALO2(K)=S
               ENDIF
1368        CONTINUE
1370     CONTINUE
         IF(ISWAP .GT. 0) THEN
            WRITE(IWR,1372) IXS(I),IYS(I),DEPO2(ISWAP)
1372        FORMAT(2X,'RECOMPUTED SIGMA AT X,Y: ',I4,',',I4,
     *             ' IS OUT OF SEQUENCE STARTING AT DEPTH: ',F5.0)
         ENDIF
      ENDIF
C
C WRITE OUTPUT DATA TO DISC
C
      IF(MOD(I,50) .EQ. 0) WRITE(IWR,1252) I,MPTS
      IF(LUSVF .GT. 0) THEN
         DO 1380 J=1,NZS
            SV(J)=SVEL(SALO2(J),TEMPO2(J),DEPO2(J))
1380     CONTINUE
         WRITE(LUSVF,REC=I) (SV(J),J=1,NZS)
      ENDIF
      IF(LUTF .GT. 0) WRITE(LUTF,REC=I) (TEMPO2(J),J=1,NZS)
      IF(LUSF .GT. 0) WRITE(LUSF,REC=I) (SALO2(J),J=1,NZS)
1390  CONTINUE
      IF(MOD(MPTS,50) .NE. 0) WRITE(IWR,1252) MPTS,MPTS
      CALL DATIME(IH,IA,ISEC)
      IF(LUSVF .GT. 0) THEN
         JBUF(41+NWFN)=IH(1)
         JBUF(42+NWFN)=IH(2)
         JBUF(43+NWFN)=IA(1)
         JBUF(44+NWFN)=IA(2)
      ENDIF
      IF(LUTF .GT. 0) THEN
         JBUF(51+NWFN)=IH(1)
         JBUF(52+NWFN)=IH(2)
         JBUF(53+NWFN)=IA(1)
         JBUF(54+NWFN)=IA(2)
      ENDIF
      IF(LUSF .GT. 0) THEN
         JBUF(61+NWFN)=IH(1)
         JBUF(62+NWFN)=IH(2)
         JBUF(63+NWFN)=IA(1)
         JBUF(64+NWFN)=IA(2)
      ENDIF
      CALL CLSFIL(LUSO,MXFS)
      WRITE(LUCF,REC=2) (JBUF(I),I=1,NWPRCF)
      GO TO 100
C
C END - END THE PROGRAM
C
9000  CLOSE(UNIT=LUCF,STATUS='KEEP')
      STOP
      END
C
C*****************************************************************************
C SUBROUTINE: INPAR
C            THIS SUBROUTINE INITIALIZES ALL THE PROCESSING PARAMETERS TO
C            DEFAULT VALUES
C IPAR(11): NUMBER OF POINTS USED FOR FRONT CURVE (NPTS)
C IPAR(12): NUMBER OF POINTS USED TO PERFORM SMOOTHING FUNCTION (LM)
C PAR(13):  RIPPLE POWER (RLP)
```

```
C IPAR(14): NUMBER OF ITERATION (ITER)
C PAR(21):  RELAXATION COEFFICIENT (ALFA)
C PAR(22):  ROSSBY DEFORMATION RADIUS (BETA)
C PAR(23):  H GRID SPACING (H)
C IPAR(24): MAXIMUM H VALUE (MAXH)
C IPAR(25): MINIMUM H VALUE (MINH)
C IPAR(26): NUMBER OF ITERATION (JTER)
C IPAR(31): LATITUDE OF DESIRED LOCATION (LAT)
C IPAR(32): LONGITUDE OF DESIRED LOCATION (LONG)
C IPAR(33): DEPTH INCREMENT (IZ)
C PAR(51):  EXPONENT VARIABLE (E)
C IPAR(52): RECOMPUTE SIMGA FIELD FLAG (IRCF)
C******************************************************************************
C
      SUBROUTINE INPAR(MXFCPS,IPAR)
      INTEGER*4 IPAR(*)
      EQUIVALENCE (IV,FV)
      DATA NO/4HN   /
C
C
      IPAR(11)=MXFCPS
      IPAR(12)=10
      FV=2.
      IPAR(13)=IV
      IPAR(14)=20
      FV=1.7
      IPAR(21)=IV
      FV=20.
      IPAR(22)=IV
      FV=.4
      IPAR(23)=IV
      IPAR(24)=100
      IPAR(25)=-100
      IPAR(26)=1000
      IPAR(31)=30
      IPAR(32)=-70
      IPAR(33)=10
      FV=1.
      IPAR(51)=IV
      IPAR(52)=NO
      RETURN
      END
C
C******************************************************************************
C SUBROUTINE: SETFS
C             THIS SUBROUTINES ALLOWS THE USER TO SET UP REQUIRED INPUT
C             AND OUTPUF FILES
C******************************************************************************
C
      SUBROUTINE SETFS(IRD,IWR,NWFN,IBUF,IH,FA,IA,IST)
      PARAMETER (NDIRS=10)
      INTEGER*4 BLANKS,IBUF(*),IH(*),IA(*),IDIRS(NDIRS)
      REAL FA(*)
      DATA BLANKS/4H    /
      DATA IDIRS/4HLD  ,4HLF  ,4HEX  ,4HEND ,4HDBF ,4HHMF ,4HSFF ,
     *           4HSVF ,4HTF  ,4HSF  /
C
C LD
C
10    WRITE(IWR,20) (IDIRS(I),I=1,NDIRS)
20    FORMAT(
     *2X,A4,'= LIST SF DIRECTIVES',/,
     *2X,A4,'= LIST FILE NAMES',/,
     *2X,A4,'= EXIT FROM DIRECTIVE SF',/,
     *2X,A4,'= END THE PROGRAM',/,
     *2X,'***** AVAILABLE FILE TYPES *****',/,
     *2X,A4,'= LEVITUS DATABASE REQUIRED BY DIRECTIVES TS AND SV ',
     *      'AS INPUT',/,
     *2X,A4,'= Q MATRIX FILE OUTPUT BY DIRECTIVE HELM',/,
```

```
     *2X,A4,'= SIGMA FILE OUTPUT BY DIRECTIVE SIG',/,
     *2X,A4,'= SOUND VELOCITY FILE OUTPUT BY DIRECTIVE SV',/,
     *2X,A4,'= BLENDED TEMPERATURE FILE OUTPUT BY DIRECTIVE SV',/,
     *2X,A4,'= BLENDED SALINITY FILE OUTPUT BY DIRECTIVE SV')
30    WRITE(IWR,40)
40    FORMAT(2X,'ENTER SF SUBDIRECTIVE NAME')
      CALL RUI(IRD,IWR,IH,FA,IA,NVS)
      DO 50 IGO=1,NDIRS
         IF(IH(1) .EQ. IDIRS(IGO)) GO TO 100
50    CONTINUE
      WRITE(IWR,60)
60    FORMAT(2X,'INVALID SF SUBDIRECTIVE')
      GO TO 30
C
C
100   IF(IGO .GT. 4) THEN
         WRITE(IWR,102) IDIRS(IGO)
102      FORMAT(2X,'ENTER ',A4,' FILE NAME')
         CALL RUI(IRD,IWR,IH,FA,IA,NVS)
      ENDIF
      GO TO (10,110,900,900,120,130,130,130,130,130),IGO
C
C LF
C
110   WRITE(IWR,112) IDIRS(5),(IBUF(I),I=1,12)
112   FORMAT(2X,A4,'= ',12A4)
      WRITE(IWR,114) IDIRS(6),(IBUF(I),I=21,20+NWFN+4)
114   FORMAT(2X,A4,'= ',6A4,T33,'LAST WRITTEN TIME: ',2A4,' ',2A4)
      WRITE(IWR,114) IDIRS(7),(IBUF(I),I=31,30+NWFN+4)
      WRITE(IWR,114) IDIRS(8),(IBUF(I),I=41,40+NWFN+4)
      WRITE(IWR,114) IDIRS(9),(IBUF(I),I=51,50+NWFN+4)
      WRITE(IWR,114) IDIRS(10),(IBUF(I),I=61,60+NWFN+4)
      GO TO 30
C
C SET DBF FILE NAME
C
120   DO 124 I=1,12
         IBUF(I)=IH(I)
124   CONTINUE
      GO TO 30
C
C SET HMF,SFF,SVF,TF,SF FILE NAME
C
130   IFLAG=0
      J=(IGO-6)*(NWFN+4)+20
      DO 132 I=1,NWFN
         IF(IBUF(J+I) .NE. IH(I)) THEN
            IBUF(J+I)=IH(I)
            IFLAG=1
         ENDIF
132   CONTINUE
      IF(IFLAG .EQ. 1) THEN
         DO 134 I=1,4
            IBUF(J+NWFN+I)=BLANKS
134      CONTINUE
      ENDIF
      GO TO 30
C
C EX, END
C
900   IF(IGO .EQ. 3) THEN
         IST=0
      ELSE
         IST=-1
      ENDIF
      RETURN
      END
C
C***************************************************************************
```

```
C SUBROUTINE: SETPAR
C             THIS SUBROUTINES ALLOWS THE USER TO SET UP ALL REQUIRED
C             PROCESSING PARAMETERS
C*******************************************************************************
C
      SUBROUTINE SETPAR(IRD,IWR,MXFCPS,IPAR,IH,FA,IA,IST)
      PARAMETER (NDIRS=19)
      INTEGER*4 YES(2),IPAR(*),IH(*),IA(*),IDIRS(NDIRS)
      REAL FA(*)
      EQUIVALENCE (IV,FV)
      DATA IDIRS/4HLD  ,4HLP  ,4HEX  ,4HEND ,4HNPTS,4HLM  ,4HRLP ,
     *           4HITER,4HALFA,4HBETA,4HF   ,4HMAXH,4HMINH,4HJTER,
     *           4HLAT ,4HLONG,4HIZ  ,4HE   ,4HRCF /
      DATA YES/4HY   ,4HYES /
C
C
      NPTS=IPAR(11)
      LM=IPAR(12)
      IV=IPAR(13)
      RLP=FV
      ITER=IPAR(14)
      IV=IPAR(21)
      ALFA=FV
      IV=IPAR(22)
      BETA=FV
      IV=IPAR(23)
      H=FV
      MAXH=IPAR(24)
      MINH=IPAR(25)
      JTER=IPAR(26)
      LAT=IPAR(31)
      LONG=IPAR(32)
      IZ=IPAR(33)
      IV=IPAR(51)
      E=FV
      IRCF=IPAR(52)
10    WRITE(IWR,20) (IDIRS(I),I=1,8)
20    FORMAT(
     *2X,A4,' = LIST SP SUBDIRECTIVES',/,
     *2X,A4,' = LIST PARAMETERS',/,
     *2X,A4,' = EXIST FROM SP DIRECTIVE',/,
     *2X,A4,' = END THE PROGRAM',/,
     *2X,'****** AVAILABLE PARAMETERS ******',/,
     *2X,A4,' = NUMBER OF POINTS USED TO GENERATE FRONT CURVE ',
     *      '(FRNT)',/,
     *2X,A4,' = NUMBER OF POINTS (-/+) TO SAMPLE (FRNT)',/,
     *2X,A4,' = RIPPLE POWER (FRNT)',/,
     *2X,A4,' = NUMBER OF ITERATIONS USED TO GENERATE FRONT CURVE ',
     *      '(FRNT)')
      WRITE(IWR,30) (IDIRS(I),I=9,NDIRS)
30    FORMAT(
     *2X,A4,' = RELAXATION COEFFICIENT (HELM)',/,
     *2X,A4,' = ROSSBY DEFORMATION RADIUS (HELM)',/,
     *2X,A4,' = H GRID SPACING (HELM)',/,
     *2X,A4,' = MAXIMUM H VALUE (HELM)',/,
     *2X,A4,' = MINIMUM H VALUE (HELM)',/,
     *2X,A4,' = NUMBER OF ITERATIONS APPLIED TO HELM SOLVER (HELM)',/,
     *2X,A4,' = LATITUDE OF DESIRED LOCATION IN DEGREES(TS)',/,
     *2X,A4,' = LONGITUDE OF DESIRED LOCATION IN DEGREES (TS)',/,
     *2X,A4,' = DEPTH INCREMENT (TS)',/,
     *2X,A4,' = EXPONENT VARIABLE (SV)',/,
     *2X,A4,' = FLAG (Y/N) INDICATING NECESSITY OF RECOMPUTING SIGMA ',
     *      'FIELD (SV)',/,
     *2X,4X,'   NEW SIGMA FIELD WILL BE SORTED IN ASCENDING ORDER',/,
     *2X,4X,'   BLENDED TEMPERATURE AND SALINITY VALUES WILL BE SORTED ',
     *      'ACCORDINGLY')
40    WRITE(IWR,50)
50    FORMAT(2X,'ENTER SP SUBDIRECTIVE NAME')
      CALL RUI(IRD,IWR,IH,FA,IA,NVS)
```

```
      DO 60 IGO=1,NDIRS
         IF(IH(1) .EQ. IDIRS(IGO)) GO TO 100
60    CONTINUE
      WRITE(IWR,70)
70    FORMAT(2X,'INVALID SP SUBDIRECTIVE')
      GO TO 40
C
C
100   IF(((IGO.GT.4 .AND. IGO.LT.NDIRS) .AND. NVS.EQ.0) .OR.
     *    (IGO.EQ.NDIRS)) THEN
         WRITE(IWR,102) IDIRS(IGO)
102      FORMAT(2X,'ENTER VALUE FOR PARAMETER ',A4)
         CALL RUI(IRD,IWR,IH,FA,IA,NVS)
      ENDIF
      GO TO (10,110,900,900,120,130,140,150,160,170,180,190,200,210,220,
     *       230,240,250,260),IGO
C
C LP
C
110   WRITE(IWR,112) IDIRS(5),NPTS,IDIRS(6),LM,IDIRS(7),RLP,
     *               IDIRS(8),ITER,IDIRS(9),ALFA,IDIRS(10),BETA,
     *               IDIRS(11),F,IDIRS(12),MAXH,IDIRS(13),MINH,
     *               IDIRS(14),JTER,IDIRS(15),LAT,IDIRS(16),LONG,
     *               IDIRS(17),IZ,IDIRS(18),E,IDIRS(19),IRCF
112   FORMAT(
     *2X,A4,' = ',I4,T31,A4,' = ',I4,/,
     *2X,A4,' = ',F10.5,T31,A4,' = ',I4,/,
     *2X,A4,' = ',F10.5,T31,A4,' = ',F10.5,/,
     *2X,A4,' = ',F10.5,/,
     *2X,A4,' = ',I4,T31,A4,' = ',I4,/,
     *2X,A4,' = ',I4,/,
     *2X,A4,' = ',I4,T31,A4,' = ',I4,/,
     *2X,A4,' = ',I4,/,
     *2X,A4,' = ',F10.5,T31,A4,' = ',A4)
      GO TO 40
C
C NPTS
C
120   IF(IA(1) .LE. 1000) THEN
         WRITE(IWR,122)
122      FORMAT(2X,'NPTS CANNOT BE < 1000')
      ELSE
         IF(IA(1) .GT. MXFCPS) THEN
            WRITE(IWR,124) MXFCPS
124         FORMAT(2X,'NPTS CANNOT BE > ',I4)
         ELSE
            NPTS=IA(1)
         ENDIF
      ENDIF
      GO TO 40
C
C LM
C
130   LM=IA(1)
      GO TO 40
C
C RLP
C
140   RLP=FA(1)
      GO TO 40
C
C ITER
C
150   ITER=IA(1)
      GO TO 40
C
C ALFA
C
160   ALFA=FA(1)
```

```
      GO TO 40
C
C BETA
C
170   BETA=FA(1)
      GO TO 40
C
C H
C
180   H=FA(1)
      GO TO 40
C
C MAXH
C
190   MAXH=IA(1)
      GO TO 40
C
C MINH
C
200   MINH=IA(1)
      GO TO 40
C
C JTER
C
210   JTER=IA(1)
      GO TO 40
C
C LAT
C
220   IF(IA(1).LT.-90 .OR. IA(1).GT.90) THEN
         WRITE(IWR,222)
222      FORMAT(2X,'LATITUDE MUST BE BETWEEN -90 AND 90 DEGREES')
      ELSE
         LAT=IA(1)
      ENDIF
      GO TO 40
C
C LONG
C
230   IF(IA(1).LT.-180 .OR. IA(1).GT.180) THEN
         WRITE(IWR,232)
232      FORMAT(2X,'LONGITUDE MUST BE BETWEEN -180 AND 180 DEGREES')
      ELSE
         LONG=IA(1)
      ENDIF
      GO TO 40
C
C IZ
C
240   IZ=IA(1)
      GO TO 40
C
C E
C
250   E=FA(1)
      GO TO 40
C
C IRCF
C
260   IRCF=IH(1)
      GO TO 40
C
C EX, END
C
900   IF(IGO .EQ. 3) THEN
         IPAR(11)=NPTS
         IPAR(12)=LM
         FV=RLP
         IPAR(13)=IV
```

```
         IPAR(14)=ITER
         FV=ALFA
         IPAR(21)=IV
         FV=BETA
         IPAR(22)=IV
         FV=H
         IPAR(23)=IV
         IPAR(24)=MAXH
         IPAR(25)=MINH
         IPAR(26)=JTER
         IPAR(31)=LAT
         IPAR(32)=LONG
         IPAR(33)=IZ
         FV=E
         IPAR(51)=IV
         IPAR(52)=IRCF
         IST=0
      ELSE
         IST=-1
      ENDIF
      RETURN
      END
C
C***************************************************************************
C SUBROUTINE: RDTS
C           THIS SUBROUTINE READS TEMPERATURE AND SALINITY PROFILES AT
C           A GIVEN LONGITUDE AND LATITUDE LOCATION FROM LEVITUS DATABASE
C***************************************************************************
C
      SUBROUTINE RDTS(LUDBF,LONG,LAT,TEMP,SAL,NWPR,XDBF)
      REAL XDBF(*),TEMP(*),SAL(*)
C
C
      I=LAT+90
      IF(LONG .LT. 0) THEN
         J=LONG+360
      ELSE
         J=LONG
      ENDIF
      J=J/5+1
      I=I/5+1
      IREC=(J-1)*144+I+72
      READ(LUDBF,REC=IREC) (XDBF(I),I=1,NWPR)
C
C CHECK FOR 0 OBERSERVATION (-999 IS INSERTED)
C
      DO 100 I=1,30
         J=(I-1)*3+1
         IF(XDBF(J) .LE. .1) THEN
            TEMP(I)=-999.
         ELSE
            TEMP(I)=XDBF(J+1)
         ENDIF
         IF(XDBF(J+90) .LE. .1) THEN
            SAL(I)=-999.
         ELSE
            SAL(I)=XDBF(J+91)
         ENDIF
100   CONTINUE
C
C
      DO 200 I=2,30
         IF(TEMP(I) .LE. -998.) TEMP(I)=TEMP(I-1)
         IF(SAL(I) .LE. -998.) SAL(I)=SAL(I-1)
200   CONTINUE
      RETURN
      END
C
C***************************************************************************
```

```
C SUBROUTINE: SIGINT
C             THIS SUBROUTINE INTEGRATES SIGMA VALUES
C*****************************************************************************
C
      SUBROUTINE SIGINT(IZ,NZS,BVF,SIG)
      REAL BVF(*),SIG(*)
C
C
      FACTOR=(2.*3.1416)**2/(3600.**2)
      G=9.8
C
C
      IDXMAX=1
      BVFMAX=BVF(IDXMAX)
      DO 10 I=2,NZS
         IF(BVF(I) .GT. BVFMAX) THEN
            IDXMAX=I
            BVFMAX=BVF(I)
         ENDIF
10    CONTINUE
C
C
      S=SIG(IDXMAX)
      RHO=S+1000.
      C=RHO/G
      BVFSUM=0.
      DO 20 I=IDXMAX-1,1,-1
         BVF2=(BVF(I)+BVF(I+1))/2.
         RN=(BVF2**2)*FACTOR
         BVFSUM=BVFSUM+RN*IZ
         SIG(I)=S-(C*BVFSUM)
20    CONTINUE
C
C
      SIG(IDXMAX)=S
C
C
      BVFSUM=0.
      DO 30 I=IDXMAX+1,NZS
         BVF2=(BVF(I)+BVF(I-1))/2.
         RN=(BVF2**2)*FACTOR
         BVFSUM=BVFSUM-RN*IZ
         SIG(I)=S-(C*BVFSUM)
30    CONTINUE
      RETURN
      END
C
C***************************************************************************
C SUBROUTINE: LINTPL
C             THIS SUBROUTINE PERFORMS LINEAR INTERPOLATION. LINEAR
C             EXTRAPOLATION ON BOTH ENDS IF REQUEST DATA IS NOT WITHIN
C             INPUT DATA RANGES
C***************************************************************************
C
      SUBROUTINE LINTPL(IWR,L,X,Y,N,U,V,INDEX)
      INTEGER*4 INDEX(*)
      REAL X(*),Y(*),U(*),V(*)
C
C
      DO 50 I=2,L
         IF(X(I-1) .EQ. X(I)) THEN
            WRITE(IWR,10)
10          FORMAT(2X,'LINTPL: IDENTICAL X VALUES')
            WRITE(IWR,20) I,X(I)
20          FORMAT(2X,'        I: ',I7,5X,'X(I) = ',E12.3)
            WRITE(IWR,30) L,N
30          FORMAT(2X,'LINTPL  L: ',I7,5X,'N    = ',I7)
            RETURN
         ELSE
```

```
         IF(X(I-1) .GT. X(I)) THEN
            WRITE(IWR,40)
40          FORMAT(2X,'LINTPL: X VALUES OUT OF SEQUENCE')
            WRITE(IWR,20) I,X(I)
            WRITE(IWR,30) L,N
            RETURN
         ENDIF
      ENDIF
50    CONTINUE
C
C
      DO 100 I=1,N
         IF(U(I) .GE. X(1)) GO TO 120
100   CONTINUE
C
C
120   ISTART=I
      DY=Y(2)-Y(1)
      DO 130 I=ISTART-1,1,-1
         V(I)=Y(1)-(ISTART-I)*DY
130   CONTINUE
C
C
      DO 140 I=1,L+L
         INDEX(I)=0
140   CONTINUE
C
C
      J=1
      DO 160 I=ISTART,N
         IBP=J
         DO 150 J=IBP,L-1
            IF(U(I).GE.X(J) .AND. U(I).LE.X(J+1)) THEN
               J2=(J-1)*2+1
               IF(INDEX(J2) .LE. 0) INDEX(J2)=I
               INDEX(J2+1)=I
               GO TO 160
            ENDIF
150      CONTINUE
160   CONTINUE
C
C
      DO 180 J=1,L-1
         J2=(J-1)*2+1
         IF(INDEX(J2) .NE. 0) THEN
            IBP=INDEX(J2)
            IEP=INDEX(J2+1)
            DY=Y(J+1)-Y(J)
            DX=X(J+1)-X(J)
            DO 170 I=IBP,IEP
               DX1=U(I)-X(J)
               R=DX1/DX
               V(I)=Y(J)+R*DY
170         CONTINUE
         ENDIF
180   CONTINUE
C
C
      DO 190 I=N,1,-1
         IF(U(!) .LE. X(L)) GO TO 200
190   CONTINUE
C
C
200   IEND=I
      DY=Y(L)-Y(L-1)
      DO 210 I=IEND+1,N
         V(I)=Y(L)+(I-IEND)*DY
210   CONTINUE
      RETURN
```

```
      END
C
C*******************************************************************************
C SUBROUTINE: INTRPL (INTERPOLATION OF A SINGLE VALUED FUNCTION)
C             THIS SUBROUTINE INTERPOLATES, FROM VALUES OF THE FUNCTION
C             GIVEN A ORDINATES OF INPUT DATA POINTS IN THE X-Y PLANE AND
C             FOR A GIVEN SET OF X VALUES (ABCISSAS), THE VALUES OF A
C             SINGLE VALUES FUNCTION Y=Y(X)
C
C AUTHOR:     HIROSHI AKIMA, U.S. DEPT. OF COMMERCE, OFFICE OF
C             TELECOMMUNICATIONS, INSTITUTE OF TELECOMMUNICATIONS SCIENCES,
C             BOULDER, COLORADO (THIS ALGORITHM WAS PUBLISHED IN COMM. ACM.
C             15(10), OCTOBER 1972
C*******************************************************************************
C
      SUBROUTINE INTRPL(IWR,L,X,Y,N,U,V)
      REAL X(*),Y(*),U(*),V(*),M1,M2,M3,M4,M5
      EQUIVALENCE (P0,X3),(Q0,Y3),(Q1,T3)
      EQUIVALENCE (UK,DX),(IMN,X2,A1,M1),(IMX,X5,A5,M5),(J,SW,SA),
     *            (Y2,W2,W4,Q2),(Y5,W3,Q3)
C
C PRELIMINARY PROCESSING
C
      L0=L
      LM1=L0-1
      LM2=LM1-1
      LP1=L0+1
      N0=N
      IF(LM2 .LT. 0) THEN
         WRITE(IWR,10)
10       FORMAT(2X,'INTRPL: L = 1 OR LESS')
         WRITE(IWR,20) L0,N0
20       FORMAT(2X,'INTRPL: L = ',I7,5X,'N = ',I7)
         RETURN
      ELSE
         IF(N0 .LE. 0) THEN
            WRITE(IWR,30)
30          FORMAT(2X,'INTRPL: N = 0 OR LESS')
            WRITE(IWR,20) L0,N0
            RETURN
         ENDIF
      ENDIF
C
C
      DO 70 I=2,L0
         IF(X(I-1) .EQ. X(I)) THEN
            WRITE(IWR,40)
40          FORMAT(2X,'INTRPL: IDENTICAL X VALUES')
            WRITE(IWR,50) I,X(I)
50          FORMAT(2X,'        I = ',I7,5X,'X(I) = ',E12.3)
            WRITE(IWR,20) L0,N0
            RETURN
         ELSE
            IF(X(I-1) .GT. X(I)) THEN
               WRITE(IWR,60)
60             FORMAT(2X,'INTRPL: X VALUES OUT OF SEQUENCE')
               WRITE(IWR,50) I,X(I)
               WRITE(IWR,20) L0,N0
               RETURN
            ENDIF
         ENDIF
70    CONTINUE
C
C
      IPV=0
      DO 900 K=1,N0
         UK=U(K)
C
C ROUTINE TO LOCATE DESIRED POINT
```

```
C
        IF(LM2 .EQ. 0) THEN
           I=2
           GO TO 300
        ELSE
           IF(UK .GE. X(LO)) THEN
              I=LP1
              GO TO 300
           ELSE
              IF(UK .LT. X(1)) THEN
                 I=1
                 GO TO 300
              ENDIF
           ENDIF
        ENDIF
        IMN=2
        IMX=LO
200     I=(IMN+IMX)/2
        IF(UK .LE. X(I)) THEN
           IMX=I
        ELSE
           IMN=I+1
        ENDIF
        IF(IMX .GT. IMN) GO TO 200
        I=IMX
C
C CHECK IF I = IPV
C
300     IF(I .NE. IPV) THEN
           IPV=I
C
C ROUTINE TO PICK UP NECESSARY X AND Y VALUES AND TO
C         ESTIMATE THEM IF NECESSARY
C
           J=I
           IF(J .EQ. 1) J=2
           IF(J .EQ. LP1) J=LO
           X3=X(J-1)
           Y3=Y(J-1)
           X4=X(J)
           Y4=Y(J)
           A3=X4-X3
           M3=(Y4-Y3)/A3
           IF(LM2 .EQ. 0) THEN
              M2=M3
           ELSE
              IF(J .NE. 2) THEN
                 X2=X(J-2)
                 Y2=Y(J-2)
                 A2=X3-X2
                 M2=(Y3-Y2)/A2
                 IF(J .EQ. LO) THEN
                    M4=M3+M3-M2
                 ELSE
                    X5=X(J+1)
                    Y5=Y(J+1)
                    A4=X5-X4
                    M4=(Y5-Y4)/A4
                 ENDIF
              ELSE
                 X5=X(J+1)
                 Y5=Y(J+1)
                 A4=X5-X4
                 M4=(Y5-Y4)/A4
                 M2=M3+M3-M4
              ENDIF
           ENDIF
           IF(J .LE. 3) THEN
              M1=M2+M2-M3
```

```
            ELSE
               A1=X2-X(J-3)
               M1=(Y2-Y(J-3))/A1
            ENDIF
            IF(J .GE. LM1) THEN
               M5=M4+M4-M3
            ELSE
               A5=X(J+2)-X5
               M5=(Y(J+2)-Y5)/A5
            ENDIF
C
C NUMERICAL DIFFERENTIATION
C
            IF(I .EQ. LP1) THEN
               W3=ABS(M5-M4)
               W4=ABS(M3-M2)
               SW=W3+W4
               IF(SW .EQ. 0.) THEN
                  W3=.5
                  W4=.5
                  SW=1.
               ENDIF
               T4=(W3*M3+W4*M4)/SW
               T3=T4
               SA=A2+A3
               T4=.5*(M4+M5-A2*(A2-A3)*(M2-M3)/(SA*SA))
               X3=X4
               Y3=Y4
               A3=A2
               M3=M4
            ELSE
               W2=ABS(M4-M3)
               W3=ABS(M2-M1)
               SW=W2+W3
               IF(SW .EQ. 0.) THEN
                  W2=.5
                  W3=.5
                  SW=1.
               ENDIF
               T3=(W2*M2+W3*M3)/SW
               IF(I .NE. 1) THEN
                  W3=ABS(M5-M4)
                  W4=^BS(M3-M2)
                  SW=W3+W4
                  IF(SW .EQ. 0.) THEN
                     W3=.5
                     W4=.5
                     SW=1.
                  ENDIF
                  T4=(W3*M3+W4*M4)/SW
               ELSE
                  T4=T3
                  SA=A3+A4
                  T3=.5*(M1+M2-A4*(A3-A4)*(M3-M4)/(SA*SA))
                  X3=X3-A4
                  Y3=Y3-M2*A4
                  A3=A4
                  M3=M2
               ENDIF
            ENDIF
C
C DETERMINATION OF THE COEFICIENTS
C
            Q2=(2.*(M3-T3)+M3-T4)/A3
            Q3=(-M3-M3+T3+T4)/(A3*A3)
         ENDIF
C
C COMPUTATION OF THE POLYNOMIAL
C
```

```
      DX=UK-P0
      V(K)=Q0+DX*(Q1+DX*(Q2+DX*Q3))
900   CONTINUE
      RETURN
      END
C
C*************************************************************************
C FUNCTION: BVFRQ
C          THIS SUBROUTINE COMPUTES BRUNT-VAISALA FREQUENCY IN CPH
C
C AUTHOR:  R. MILLARD, WOODS HOLE OCEANOGRAPHIC INSTITUTION
C
C NOTES:   USES 1980 EQUATION OF STATE
C          UNITS:
C          PRESSURE        P0     DECIBARS
C          TEMPERATURE     T      DEQ CELSIUS (IPTS-68)
C          SALINITY        S      (IPSS-78)
C          BOUYANCY FREQ   BVFRQ  CPH
C          N**2            E      RADIANS/SECOND
C
C CHECKVALUE: BVFRQ=14.57836  CPH E=6.4739928E-4 RAD/SEC.
C             S(1)=35.0, T(1)=5.0,  P(1)=1000.0
C             S(2)=35.0, T(2)=4.0,  P(2)=1000.0
C             RESULT CENTERED AT PAV=1001.0 DBARS
C             COMPUTES N IN CYCLES PER HOUR AND E=N**2 IN RAD/SEC**2
C*************************************************************************
C
      REAL FUNCTION BVFRQ(S,T,P,NOBS,PAV,E)
      REAL P(*),T(*),S(*)
C
C
      E=0.
      BVFRQ=0.
      IF(NOBS .LT. 2) RETURN
      CXX=0.
      CX=0.
      CXY=0.
      CY=0.
C
C COMPUTE LEAST SQUARES ESTIMATE OF SPECIFIC VOLUME ANAMOLY GRADIENT
C
      DO 10 K=1,NOBS
         CX=CX+P(K)
10    CONTINUE
C
C
      PAV=CX/NOBS
      DO 20 K=1,NOBS
         DATA=SVAN(S(K),THETA(S(K),T(K),P(K),PAV),PAV,SIG)*1.0E-8
         CXY=CXY+DATA*(P(K)-PAV)
         CY=CY+DATA
         CXX=CXX+(P(K)-PAV)**2
20    CONTINUE
C
C
      IF(CXX .EQ. 0.) RETURN
      A0=CXY/CXX
      V350P=(1./(SIG+1000.))-DATA
      VBAR=V350P+CY/NOBS
      DVDP=A0
C
C
      IF(VBAR .EQ. 0.) RETURN
      E=-.96168423E-2*DVDP/(VBAR)**2
      BVFRQ=572.9578*SIGN(SQRT(ABS(E)),E)
      RETURN
      END
C
C*************************************************************************
```

```
C FUNCTION: SVAN
C          SPECIFIC VOLUME ANOMALY (STERIC ANOMALY) BASED ON 1980 EQUATION
C          OF STATE FOR SEAWATER AND 1978 PRACTICAL SALINITY SCALE
C
C REFERENCE:MILLERO, ET AL (1980) DEEP-SEA RES., 27A, PP 255-264
C          MILLERO AND POISSON 1981, DEEP-SEA RES., 28A, PP 625-629
C          BOTH ABOVE REFERENCES ARE ALSO FOUND IN UNESCO REPORT 38 (1981)
C
C          UNITS:
C          PRESSURE          PO     DECIBARS
C          TEMPERATURE       T      DEG CELSIUS (IPTS-68)
C          SALINITY          S      (IPSS-78)
C          SPEC. VOL. ANA    SVAN   M**3/KG*1.0E-8
C          DENSITY ANA.      SIGMA  KG/M**3
C
C CHECKVALUE: SVAN=981.3021 E-8 M**3/KG  FOR S=40 (IPSS-78)
C             T=40 DEG C, PO=10000 DECIBARS
C
C             SIGMA=59.82037 KG/M**3 FOR S=40 (IPSS-78)
C             T=40 DEG C, PO=10000 DECIBARS
C
C NOTE:       R4 IS REFERED TO AS C IN MILLERO AND POISSON 1981
C***************************************************************************
C
      REAL FUNCTION SVAN(S,T,PO,SIGMA)
      REAL K,KO,KW,K35
      EQUIVALENCE (E,D,B1),(BW,B,R3),(C,A1,R2),(AW,A,R1),(KW,KO,K)
      DATA R3500/1028.1063/
      DATA R4/4.8314E-4/
      DATA DR350/28.106331/
C
C CONVERT PRESSURE TO BARS AND TAKE SQUARE ROOT SALINITY
C
      P=PO/10.
      SR=SQRT(ABS(S))
C
C PURE WATER DENSITY AT ATMOSPHERIC PRESSURE
C     BIGG P.H., (1967) BR. J. APPLIED PHYSICS 8, PP 521-537
C
      R1=((((6.536332E-9*T-1.120083E-6)*T+1.001685E-4)*T
     *   -9.095290E-3)*T+6.793952E-2)*T-28.263737
C
C SEAWATER DENSITY ATM PRESS
C COEFFICIENTS INVOLVING SALINITY
C R2= A IN NOTATION OF MILLERO AND POISSON 1981
C
      R2=(((5.3875E-9*T-8.2467E-7)*T+7.6438E-5)*T-4.0899E-3)*T
     *   +8.24493E-1
C
C R3=B IN NOTATION OF MILLERO AND POISSON 1981
C
      R3=(-1.6546E-6*T+1.0227E-4)*T-5.72466E-3
C
C INTERNATIONAL ONE-ATMOSPHERE EQUATION OF STATE OF SEAWATER
C
      SIG=(R4*S+R3*SR+R2)*S+R1
C
C SPECIFIC VOLUME AT ATMOSPHERIC PRESSURE
C
      V350P=1./R3500
      SVA=-SIG*V350P/(R3500+SIG)
      SIGMA=SIG+DR350
C
C SCALE SPECIFIC VOL. ANAMOLY TO NORMALLY REPORTED UNITS
C
      SVAN=SVA*1.0E+8
      IF(P .EQ. 0.) RETURN
C
C NEW HIGH PRESSURE EQUATION OF STATE FOR SEAWATER
```

```
C     MILLERO, ET AL, 1980 DSR 27A, PP 255-264
C     CONSTANT NOTATION FOLLOWS ARTICLE
C COMPUTE COMPRESSION TERMS
C
      E=(9.1697E-10*T+2.0816E-8)*T-9.9348E-7
      BW=(5.2787E-8*T-6.12293E-6)*T+3.47718E-5
      B=BW+E*S
      D=1.91075E-4
      C=(-1.6078E-6*T-1.0981E-5)*T+2.2838E-3
      AW=((-5.77905E-7*T+1.16092E-4)*T+1.43713E-3)*T-.1194975
      A=(D*SR+C)*S+AW
      B1=(-5.3009E-4*T+1.6483E-2)*T+7.944E-2
      A1=((-6.1670E-5*T+1.09987E-2)*T-0.603459)*T+54.6746
      KW=(((-5.155288E-5*T+1.360477E-2)*T-2.327105)*T+148.4206)*T
     *   -1930.06
      K0=(B1*SR+A1)*S+KW
C
C EVALUATE PRESSURE POLYNOMIAL
C          K EQUALS THE SECANT BULK MODULUS OF SEAWATER
C          DK=K(S,T,P)-K(35,0,P)
C          K35=K(35,0,P)
C
      DK=(B*P+A)*P+K0
      K35=(5.03217E-5*P+3.359406)*P+21582.27
      GAM=P/K35
      PK=1.-GAM
      SVA=SVA*PK+(V350P+SVA)*P*DK/(K35*(K35+DK))
C
C SCALE SPECIFIC VOL. ANAMOLY TO NORMALLY REPORTED UNITS
C
      SVAN=SVA*1.0E+8
      V350P=V350P*PK
C
C COMPUTE DENSITY ANAMOLY WITH RESPECT TO 1000.0 KG/M**3
C         1. DENSITY ANAMOLY AT 35 (IPSS-78), 0 DEG. C. AND 0 DECIBARS
C         2. DR35P: DENSITY ANAMOLY 35 (IPSS-78), 0 DEG. C., PRES. VARIATION
C         3. DVAN: DENSITY ANAMOLY VARIATIONS INVOLVING SPECIFIC VOL. ANAMOLY
C
C CHECK VALUE: SIGMA=59.82037 KG/M**3 FOR S=40 (IPSS-78),
C              T=40 DEG. C., P0=10000 DECIBARS
C
      DR35P=GAM/V350P
      DVAN=SVA/(V350P*(V350P+SVA))
      SIGMA=DR350+DR35P-DVAN
      RETURN
      END
C
C***********************************************************************
C FUNCTION: THETA
C          THIS FUNCTION COMPUTES POTENTIAL TEMPERATURE AT PR USING
C          BRYDEN 1973 POLYNOMIAL FOR ADIABATIC LAPSE RATE AND
C          RUNGE-KUTTA 4-TH ORDER INTEGRATION ALGORITHM.
C
C REFERENCE: BRYDEN, H., 1973, DEEP-SEA RES., 20, PP 401-408
C            FOFONOFF, N. 1977, DEEP-SEA RES., 24, PP 489-491
C
C            UNITS:  PRESSURE        P0    DECIBARS
C                    TEMPERATURE     T0    DEG CELSIUS (IPTS-68)
C                    SALINITY        S     (IPSS-78)
C                    REFERENCE PRS   PR    DECIBARS
C                    POTENTIAL TMP   THETA DEG CELSIUS
C
C CHECKVALUE: THETA=36.89073 C, S=40 (IPSS-78), T0=40 DEG. C.,
C             P0=10000 DECIBARS, PR=0 DECIBARS
C***********************************************************************
C
      REAL FUNCTION THETA(S,T0,P0,PR)
C
C SET UP INTERMEDIATE TEMPERATURE AND PRESSURE VARIABLES
```

```
C
      P=PO
      T=TO
      H=PR-P
      XK=H*ATG(S,T,P)
      T=T+.5*XK
      Q=XK
      P=P+.5*H
      XK=H*ATG(S,T,P)
      T=T+.29289322*(XK-Q)
      Q=.58578644*XK+.121320344*Q
      XK=H*ATG(S,T,P)
      T=T+1.707106781*(XK-Q)
      Q=3.414213562*XK-4.121320344*Q
      P=P+.5*H
      XK=H*ATG(S,T,P)
      THETA=T+(XK-2.*Q)/6.
      RETURN
      END
C
C******************************************************************************
C FUNCTION: ATG
C           ADIABATIC TEMPERATURE GRADIENT DEG. C. PER DECIBAR
C
C REFERENCE: BRYDEN, H., 1973 DEEP-SEA RES., 20, PP 401-408
C
C           UNITS: PRESSURE     P     DECIBARS
C                  TEMPERATURE  T     DEG CELSIUS (IPTS-68)
C                  SALINITY     S     (IPSS-78)
C                  ADIABATIC    ATG   DEG. C/DECIBAR
C
C CHECKVALUE: ATG=3.255976E-4 C/DBAR FOR S=40 (IPSS-78)
C             T=40 DEG. C., PO=10000 DECIBARS
C******************************************************************************
C
      REAL FUNCTION ATG(S,T,P)
C
C
      DS=S-35.
      ATG=(((-2.1687E-16*T+1.8676E-14)*T-4.6206E-13)*P
     *    +((2.7759E-12*T-1.1351E-10)*DS+((-5.4481E-14*T
     *    +8.733E-12)*T-6.7795E-10)*T+1.8741E-8))*P
     *    +(-4.2393E-8*T+1.8932E-6)*DS
     *    +((6.6228E-10*T-6.836E-8)*T+8.5258E-6)*T+3.5803E-5
      RETURN
      END
C
C******************************************************************************
C FUNCTION: SVEL - COMPUTING SOUND VELOCITY
C
C REFERENCE: SOUND SPEED SEAWATER CHEN AND MILLERO 1977, JASA, 62,
C            PP 1129 - 1135
C
C UNITS:     PRESSURE       PO    DECIBARS
C            TEMPERATURE    T     DEG. CELSIUS (IPTS-68)
C            SALINITY       S     (IPSS-78)
C            SOUND SPEED    SVEL  METERS/SECOND
C
C CHECKVALUE: SVEL = 1.731.995 M/S, S=40 (IPSS-78),
C                T = 40 DEG C
C                P = 10000 DBAR
C******************************************************************************
C
      REAL FUNCTION SVEL(S,T,PO)
      EQUIVALENCE (A0,B0,C0),(A1,B1,C1),(A2,C2),(A3,C3)
C
C SCALE PRESSURE TO BARS
C
      P=PO/10.
```

```
      SR=SQRT(ABS(S))
C
C S**2 TERM
C
      D=1.727E-3-7.9836E-6*P
C
C S**3/2 TERM
C
      B1=7.3637E-5+1.7945E-7*T
      B0=-1.922E-2-4.42E-5*T
      B=B0+B1*P
C
C S**1 TERM
C
      A3=(-3.389E-13*T+6.649E-12)*T+1.100E-10
      A2=((7.988E-12*T-1.6002E-10)*T+9.1041E-9)*T-3.9064E-7
      A1=(((-2.0122E-10*T+1.0507E-8)*T-6.4885E-8)*T-1.2580E-5)*T
     *   +9.4742E-5
      A0=(((-3.21E-8*T+2.006E-6)*T+7.164E-5)*T-1.262E-2)*T+1.389
      A=((A3*P+A2)*P+A1)*P+A0
C
C S**0 TERM
C
      C3=(-2.3643E-12*T+3.8504E-10)*T-9.7729E-9
      C2=(((1.0405E-12*T-2.5335E-10)*T+2.5974E-8)*T-1.7107E-6)*T
     *   +3.1260E-5
      C1=(((-6.1185E-10*T+1.3621E-7)*T-8.1788E-6)*T+6.8982E-4)*T
     *   +.153563
      C0=((((3.1464E-9*T-1.47800E-6)*T+3.3420E-4)*T-5.80852E-2)*T
     *   +5.03711)*T+1402.388
      C=((C3*P+C2)*P+C1)*P+C0
C
C SOUND SPEED
C
      SVEL=C+(A+B*SR+D*S)*S
      RETURN
      END
```

```
      A0=(((-3.21E-8*T+2.006E-6)*T+7.164E-5)*T-1.262E-2)*T+1.389
      A=((A3*P+A2)*P+A1)*P+A0
C
C S**0 TERM
C
      C3=(-2.3643E-12*T+3.8504E-10)*T-9.7729E-9
      C2=(((1.0405E-12*T-2.5335E-10)*T+2.5974E-8)*T-1.7107E-6)*T
     *   +3.1260E-5
      C1=(((-6.1185E-10*T+1.3621E-7)*T-8.1788E-6)*T+6.8982E-4)*T
     *   +.153563
      C0=((((3.1464E-9*T-1.47800E-6)*T+3.3420E-4)*T-5.80852E-2)*T
     *   +5.03711)*T+1402.388
      C=((C3*P+C2)*P+C1)*P+C0
C
C SOUND SPEED
C
      SVEL=C+(A+B*SR+D*S)*S
      RETURN
      END
```

```
C
C*****************************************************************************
C SUBROUTINE: CKFILE
C           THIS SUBROUTINE CHECKS EXISTENCE AND RECORD SIZE OF A GIVEN
C           FILE NAME
C*****************************************************************************
C
      SUBROUTINE CKFILE(IWR,ITYPE,FILNAM,IWF,LUS,LUSO,FILTYP,NWPR,IST)
      CHARACTER*(*) FILNAM
      LOGICAL FEXIST
      INTEGER*4 LUS(*),LUSO(*),FILTYP(*)
C
C
      NBPR=NWPR*4
      INQUIRE(FILE=FILNAM,EXIST=FEXIST,RECL=NBYTES)
      GO TO (100,200),ITYPE
C
C CHECK FOR A FILE WHICH MUST CURRENTLY EXIST
C
100   IF(FEXIST) THEN
         IF(NBYTES .NE. NBPR) THEN
            WRITE(IWR,110) FILTYP(IWF)
110         FORMAT(2X,'WRONG FILE IS DESIGNATED AS ',A4,' FILE')
            IST=-1
         ELSE
            LU=LUS(IWF)
            LUSO(IWF)=LU
            IF(IWF .EQ. 3) THEN
               WRITE(IWR,120) FILTYP(IWF),FILNAM
120            FORMAT(2X,'OPEN EXISTING ',A4,' FILE: ',A48)
               OPEN(UNIT=LU,FILE=FILNAM,STATUS='OLD',FORM='UNFORMATTED',
     *              ACCESS='DIRECT',RECL=NWPR,READONLY)
            ELSE
               WRITE(IWR,130) FILTYP(IWF),FILNAM
130            FORMAT(2X,'OPEN EXISTING ',A4,' FILE: ',A24)
               OPEN(UNIT=LU,FILE=FILNAM,STATUS='OLD',FORM='UNFORMATTED',
     *              ACCESS='DIRECT',RECL=NWPR)
            ENDIF
            IST=0
         ENDIF
      ELSE
         IF(IWF .EQ. 3) THEN
            WRITE(IWR,140) FILTYP(IWF),FILNAM
140         FORMAT(2X,A4,' FILE: ',A48,' DOES NOT EXIST')
         ELSE
            WRITE(IWR,150) FILTYP(IWF),FILNAM
150         FORMAT(2X,A4,' FILE: ',A24,' DOES NOT EXIST')
         ENDIF
         IST=-1
      ENDIF
      RETURN
C
C CHECK FOR A FILE WHICH MAY NOT CURRENTLY EXIST
C       ALLOCATE THE FILE IF IT DOES NOT EXIST
C
200   IF(FEXIST) THEN
         IF(NBYTES .NE. NBPR) THEN
            WRITE(IWR,110) FILTYP(IWF)
            IST=-1
         ELSE
            LU=LUS(IWF)
            LUSO(IWF)=LU
            WRITE(IWR,130) FILTYP(IWF),FILNAM
            OPEN(UNIT=LU,FILE=FILNAM,STATUS='OLD',FORM='UNFORMATTED',
     *           ACCESS='DIRECT',RECL=NWPR)
            IST=0
         ENDIF
      ELSE
         LU=LUS(IWF)
```

```
         LUSO(IWF)=LU
         WRITE(IWR,220) FILTYP(IWF),FILNAM
220      FORMAT(2X,'OPEN A NEW ',A4,' FILE: ',A24)
         OPEN(UNIT=LU,FILE=FILNAM,STATUS='NEW',FORM='UNFORMATTED',
     *        ACCESS='DIRECT',RECL=NWPR)
         IST=1
      ENDIF
      RETURN
      END
C
C*************************************************************************
C SUBROUTINE: CLSFIL
C            THIS SUBROUTINE CLOSES FILES IF THEY ARE CURRENTLY OPEN
C*************************************************************************
C
      SUBROUTINE CLSFIL(LUSO,MXFS)
      INTEGER*4 LUSO(*)
C
C
      DO 10 I=2,MXFS
         IF(LUSO(I) .GT. 0) THEN
            CLOSE(UNIT=LUSO(I),STATUS='KEEP')
            LUSO(I)=-1
         ENDIF
10    CONTINUE
      RETURN
      END
C
C*************************************************************************
C SUBROUTINE: RUI
C           THIS SUBROUTINE READS USER'S INPUT AS CHARACTER STRING AND
C           THEN CONVERTS EACH SUBSTRING INTO HOLLERITH, INTEGER OR
C           FLOATING VALUES ACCORDINGLY
C*************************************************************************
C
      SUBROUTINE RUI(IRD,IWR,IH,FA,IA,NVS)
      INTEGER*2 IBUF(3,40)
      INTEGER*4 BLANKS,PERIOD,IH(*),IA(*)
      REAL FA(*),XBUF(40)
      CHARACTER*80 CHRBUF,CA
      SAVE IC,NCS,CHRBUF
      DATA IC,NCS/0,0/
      DATA BLANKS/4H    /
C
C
      IF(NCS .EQ. 0 .OR. IC .GT. NCS) THEN
         PERIOD=ICHAR('.')
         MINUS=ICHAR('-')
         IZERO=ICHAR('0')
         NINE=ICHAR('9')
         WRITE(IWR,10)
10       FORMAT(' >',$)
         READ(IRD,20) CHRBUF
20       FORMAT(A80)
C
C CONVERT ALPHABETIC CHARACTERS FROM LOWER CASE TO UPPER CASE
C
         DO 30 I=1,80
            IV=ICHAR(CHRBUF(I:I))
            IF(IV.GE.97 .AND. IV.LE.122) THEN
               IV=IV-32
               CHRBUF(I:I)=CHAR(IV)
            ENDIF
30       CONTINUE
C
C FIND ALL SUBSTRINGS
C
         IC=1
         NCS=0
```

```
      IB=0
      IE=0
      DO 50 I=1,80
         IF(CHRBUF(I:I).NE.' ' .AND. CHRBUF(I:I).NE.',') THEN
            IF(IB .EQ. 0) IB=I
            IE=I
         ELSE
            IF(IB .GT. 0) THEN
               NCS=NCS+1
               IBUF(1,NCS)=IB
               IBUF(2,NCS)=IE
               IBUF(3,NCS)=0
               XBUF(NCS)=0.
               IB=0
               IE=0
            ENDIF
         ENDIF
50    CONTINUE
      IF(IB .GT. 0) THEN
         NCS=NCS+1
         IBUF(1,NCS)=IB
         IBUF(2,NCS)=IE
         IBUF(3,NCS)=0
         XBUF(NCS)=0.
      ENDIF
C
C CONVERT NUMERIC SUBSTRINGS INTO FLOATING-POINT VALUES
C
      DO 90 I=1,NCS
         IB=IBUF(1,I)
         IE=IBUF(2,I)
         IV=ICHAR(CHRBUF(IB:IB))
         IF(IV .EQ. MINUS) THEN
            IB=IB+1
            ISIGN=-1
         ELSE
            ISIGN=1
         ENDIF
         XF1=0.
         XF2=0.
         IFLAG=0
         NDGTS=0
         DO 60 J=IB,IE
            IV=ICHAR(CHRBUF(J:J))
            IF((IV.LT.IZERO .OR. IV.GT.NINE) .AND.
     *         IV.NE.PERIOD) GO TO 90
            IF(IV .EQ. PERIOD) THEN
               IFLAG=1
            ELSE
               IF(IFLAG .EQ. 0) THEN
                  XF1=XF1*10.+(IV-IZERO)
               ELSE
                  XF2=XF2*10.+(IV-IZERO)
                  NDGTS=NDGTS+1
               ENDIF
            ENDIF
60       CONTINUE
         IBUF(3,I)=1
         XBUF(I)=ISIGN*(XF1+(XF2/(10**NDGTS)))
90    CONTINUE
    ENDIF
C
C INITIALIZE RETURNED ARGUMENTS
C IH:  ARRAY STORED HOLLERITH VALUES
C IA:  ARRAY STORED INTEGER VALUES
C FA:  ARRAY STORED FLOATING-POINT VALUES
C NVS: NUMBER OF VALUES RETURNED IN IA,FA ARRAYS
C
    DO 100 I=1,20
```

```
         IH(I)=BLANKS
100   CONTINUE
      DO 110 I=1,10
         IA(I)=0
         FA(I)=0.
110   CONTINUE
      NVS=0
C
C RETURN PROPER HOLLERITH AND NUMERIC VALUES
C
      IF(NCS .GT. 0) THEN
         IF(IBUF(3,IC) .EQ. 0) THEN
            IB=IBUF(1,IC)
            IE=IBUF(2,IC)
            CA(1:80)=' '
            I=IE-IB+1
            CA(1:I)=CHRBUF(IB:IE)
            READ(CA,120) (IH(I),I=1,20)
120         FORMAT(20A4)
            IC=IC+1
         ENDIF
C
C RETURN ALL NUMERIC VALUES BEFORE NEXT HOLLERITH STRING IS ENCOUNTERED
C
         DO 130 I=IC,NCS
            IF(IBUF(3,I) .EQ. 0) GO TO 140
            NVS=NVS+1
            FA(NVS)=XBUF(I)
            IA(NVS)=FA(NVS)
130      CONTINUE
140      IC=I
      ENDIF
      RETURN
      END
C
C***************************************************************************
C SUBROUTINE: DLXY
C             THIS SUBROUTINE DETERMINES X,Y COORDINATES OF POINTS
C             BETWEEN TWO GIVEN END POINTS
C***************************************************************************
C
      SUBROUTINE DLXY(IX1,IY1,IX2,IY2,NPTS,MXVS,IXS,IYS)
      INTEGER*4 IXS(*),IYS(*)
C
C
      IF(IY1 .EQ. IY2) THEN
         MINX=MIN0(IX1,IX2)
         MAXX=MAX0(IX1,IX2)
         NPTS=0
         DO 10 I=MINX,MAXX
            NPTS=NPTS+1
            IF(NPTS .GT. MXVS) RETURN
            IXS(NPTS)=I
            IYS(NPTS)=IY1
10       CONTINUE
      ELSE
         IF(IX1 .EQ. IX2) THEN
            MINY=MIN0(IY1,IY2)
            MAXY=MAX0(IY1,IY2)
            NPTS=0
            DO 20 I=MINY,MAXY
               NPTS=NPTS+1
               IF(NPTS .GT. MXVS) RETURN
               IXS(NPTS)=IX1
               IYS(NPTS)=I
20          CONTINUE
         ELSE
            YY=IY2-IY1
            XX=IX2-IX1
```

```
            SLOPE=YY/XX
            MINY=MIN0(IY1,IY2)
            MAXY=MAX0(IY1,IY2)
            IF(SLOPE .EQ. -1. .OR. SLOPE.EQ.1.) THEN
               NPTS=0
               DO 30 I=MINY,MAXY
                  Y=I-IY1
                  IX=Y/SLOPE+IX1+.5
                  NPTS=NPTS+1
                  IF(NPTS .GT. MXVS) RETURN
                  IXS(NPTS)=IX
                  IYS(NPTS)=I
30             CONTINUE
            ELSE
               MINX=MIN0(IX1,IX2)
               MAXX=MAX0(IX1,IX2)
               NPTS=1
               IF(MINY .EQ. IY1) THEN
                  IXS(NPTS)=IX1
                  IYS(NPTS)=IY1
               ELSE
                  IXS(NPTS)=IX2
                  IYS(NPTS)=IY2
               ENDIF
               DO 90 I=MINY,MAXY
                  Y=I-IY1
                  X=Y/SLOPE+IX1
                  IX=(1./SLOPE)+X+.5
                  IX=X+.5
                  JX=IXS(NPTS)
                  IF(IX .GT. JX) THEN
                     DO 40 J=JX,IX
                        IF(J.NE.IXS(NPTS) .OR. I.NE.IYS(NPTS)) THEN
                           NPTS=NPTS+1
                           IF(NPTS .GT. MXVS) RETURN
                           IXS(NPTS)=J
                           IYS(NPTS)=I
                        ENDIF
40                   CONTINUE
                  ELSE
                     DO 50 J=JX,IX,-1
                        IF(J.NE.IXS(NPTS) .OR. I.NE.IYS(NPTS)) THEN
                           NPTS=NPTS+1
                           IF(NPTS .GT. MXVS) RETURN
                           IXS(NPTS)=J
                           IYS(NPTS)=I
                        ENDIF
50                   CONTINUE
                  ENDIF
90             CONTINUE
            ENDIF
         ENDIF
      ENDIF
      RETURN
      END
C
C**********************************************************************
C SUBROUTINE: DATIME
C             THIS SUBROUTINE CALLS SYSTEM DEPENDANT ROUTINES, IDATE AND
C             TIME, TO GET CURRENT DATE AND TIME OF DAY
C**********************************************************************
C
      SUBROUTINE DATIME(DATE,ITIME,ISEC)
      INTEGER*4 IMDY(3),IHMS(3),ITEMP(2),DATE(*),ITIME(*)
      EQUIVALENCE (IMDY(1),IHMS(1))
C
C
      CALL IDATE(IMDY(1),IMDY(2),IMDY(3))
      ENCODE(8,10,ITEMP) IMDY
```

```
10    FORMAT(I2.2,'/',I2.2,'/',I2.2)
      DATE(1)=ITEMP(1)
      DATE(2)=ITEMP(2)
      CALL TIME(ITEMP)
      ITIME(1)=ITEMP(1)
      ITIME(2)=ITEMP(2)
C
C CONVERT TIME INTO SECONDS
C
      DECODE(8,20,ITEMP) IHMS
20    FORMAT(I2,1X,I2,1X,I2)
      ISEC=IHMS(1)*3600+IHMS(2)*60+IHMS(3)
      RETURN
      END
```

FSP PROGRAM LISTING

```
C
C*******************************************************************************
C PROGRAM NAME: FSP - PLOT SFF, SVF, TF OR SF FILE OUTPUT BY PROGRAM FSM
C DATE: AUGUST 7, 1990
C PROGRAMMER: TIGER CHENG (SVERDRUP)
C*******************************************************************************
C
      PROGRAM FSP
      PARAMETER (NWFN=6)
      PARAMETER (NDIRS=6)
      PARAMETER (MXFS=6)
      PARAMETER (MXPARS=6)
      PARAMETER (NBPRCF=400,NWPRCF=NBPRCF/4)
      PARAMETER (MXROWS=480,MXCOLS=640)
      PARAMETER (MXVS=MXROWS+MXCOLS)
      PARAMETER (MXWWS=90000)
      INTEGER*4 BLANKS,IH(20),IA(10),IHFN(15),
     *          FILTYP(MXFS),LUS(MXFS),LUSO(MXFS),IDIRS(NDIRS),
     *          IXS(MXVS),IYS(MXVS),INDEX(MXVS),
     *          IPAR(MXPARS),DEFPAR(MXPARS),
     *          IBUF(NWPRCF),JBUF(NWPRCF)
      REAL FA(10),PBUF(MXVS*MXVS),VBUF(MXVS)
      CHARACTER*24 CFN,PFN
      CHARACTER*60 CHFN
      EQUIVALENCE (IV,FV)
      COMMON WORK(MXWWS)
      DATA IRD/5/
      DATA IWR/6/
      DATA BLANKS/4H    /
      DATA FILTYP/4HCF  ,4HHMF ,4HSFF ,4HSVF ,4HTF  ,4HSF  /
      DATA LUS/MXFS*4/
      DATA LUSO/MXFS*-1/
      DATA IDCF/56789/
      DATA IDIRS/4HLD  ,4HIN  ,4HSP  ,4HPLOT,4HSF  ,4HEND /
C
C
      WRITE(IWR,10)
10    FORMAT(2X,'FRONT SIMULATION PLOTTING PROGRAM')
      WRITE(IWR,20)
20    FORMAT(2X,'ENTER CONTROL FILE (CF) NAME')
      CALL RUI(IRD,IWR,IH,FA,IA,NVS)
      IF(IH(1) .EQ. BLANKS) STOP
      WRITE(CFN,30) (IH(I),I=1,NWFN)
30    FORMAT(6A4)
C
C CHECK EXISTENCE OF CONTROL FILE
C
      CALL CKFILE(IWR,1,CFN,1,LUS,LUSO,FILTYP,NWPRCF,IST)
      IF(IST .NE. 0) STOP
      READ(LUS(1),REC=1) (IBUF(I),I=1,NWPRCF)
      IF(IBUF(1) .NE. IDCF) THEN
         WRITE(IWR,40)
40       FORMAT(2X,'WRONG CONTROL FILE IS USED')
         GO TO 900
      ENDIF
C
C READ IN EXTERNAL FILE NAMES STORED IN THE CONTROL FILE
C
      READ(LUS(1),REC=2) (JBUF(I),I=1,NWPRCF)
      CLOSE(UNIT=LUS(1),STATUS='KEEP')
C
C GET FSM PROCSSING PARAMETERS (IZ,NZS,MPTS)
C
      IX1=IBUF(41)
      IY1=IBUF(42)
      IX2=IBUF(43)
      IY2=IBUF(44)
```

```
      CALL DLXY(IX1,IY1,IX2,IY2,MPTS,MXVS,IXS,IYS)
      IZ=IBUF(49)
      NZS=IBUF(50)
C
C USER SELECTS A FILE TO PLOT
C
50    CALL SELECF(IRD,IWR,FILTYP,MXFS,JBUF,NWFN,IWF,IH,FA,IA)
      IF(IWF .EQ. 0) GO TO 900
C
C SET PARAMETERS TO DEFAULT VALUES BASED UPON THE SELECTED FILE
C
      FV=14.
      DEFPAR(1)=IV
      FV=11.
      DEFPAR(2)=IV
      DEFPAR(3)=1
      DEFPAR(5)=1
      IF(IWF .EQ. 2) THEN
         NWPR=MXROWS
         DEFPAR(4)=MXROWS
         DEFPAR(6)=MXCOLS
      ELSE
         NWPR=NZS
         DEFPAR(4)=NZS
         DEFPAR(6)=MPTS
      ENDIF
C
C INITIALIZE PARAMETERS TO DEFAULT VALUES
C
      DO 60 I=1,MXPARS
         IPAR(I)=DEFPAR(I)
60    CONTINUE
C
C OPEN SELECTED FILE
C
      IF(LUSO(IWF) .EQ. -1) THEN
         CALL CLSFIL(LUSO,MXFS)
         IW=(IWF-2)*10+21
         WRITE(PFN,30) (JBUF(I),I=IW,IW+NWFN-1)
         CALL CKFILE(IWR,1,PFN,IWF,LUS,LUSO,FILTYP,NWPR,IST)
         IF(IST .NE. 0) STOP
C
C ENCODE FILE NAME TO BE USED FOR PLOT HEADING
C
         CALL DATIME(IH,IA,ISEC)
         ENCODE(60,70,IHFN) FILTYP(IWF),(JBUF(I),I=IW,IW+NWFN-1),
     *                         IH(1),IH(2),IA(1),IA(2)
70       FORMAT(A4,'FILENAME: ',6A4,'    ',2A4,' ',2A4,'$')
      ENDIF
      IGO=0
      GO TO 300
C
C
100   WRITE(IWR,120)
120   FORMAT(2X,'ENTER PROGRAM MAIN DIRECTIVE NAME')
      CALL RUI(IRD,IWR,IH,FA,IA,NVS)
      DO 130 IGO=1,NDIRS
         IF(IH(1) .EQ. IDIRS(IGO)) GO TO 190
130   CONTINUE
      WRITE(IWR,140)
140   FORMAT(2X,'INVALID PROGRAM MAIN DIRECTIVE')
      GO TO 100
C
C
190   GO TO (200,300,400,500,50,900),IGO
C
C LD - LIST PROGRAM DIRECTIVES
C
200   WRITE(IWR,210)
```

```
210   FORMAT(
     *2X,'LD   - LIST PROGRAM DIRECTIVES',/,
     *2X,'IN   - INITIALIZE PLOTTING PARAMETERS',/,
     *2X,'SP   - SET PLOTTING PARAMETERS',/,
     *2X,'PLOT - GENERATE A PLOT FILE (POPFIL.DAT)',/,
     *2X,'SF   - RETURN TO FILE SELECTION MODE',/,
     *2X,'END  - END THE PROGRAM')
      GO TO 100
C
C IN - INITIALIZE PLOTTING PARAMETERS
C       IPAR(1) = PAGE X
C       IPAR(2) = PAGE Y
C       IPAR(3) = INITIAL COLUMN TO PLOT (Y AXIS)
C       IPAR(4) = LAST COLUMN TO PLOT (Y AXIS)
C       IPAR(5) = INITIAL ROW TO PLOT (X AXIS)
C       IPAR(6) = LAST ROW TO PLOT (X AXIS)
C
300   DO 310 I=1,MXPARS
         IPAR(I)=DEFPAR(I)
310   CONTINUE
      GO TO 100
C
C SP - SET PARAMETERS
C
400   CALL SETPAR(IRD,IWR,IPAR,DEFPAR,FILTYP(IWF),IH,FA,IA,IST)
      IF(IST .EQ. -1) GO TO 900
      GO TO 100
C
C PLOT
C
500   IV=IPAR(1)
      WID=FV
      IV=IPAR(2)
      HT=FV
      IY=IPAR(3)
      LY=IPAR(4)
      IX=IPAR(5)
      LX=IPAR(6)
      IXSIZE=LX-IX+1
      IYSIZE=LY-IY+1
      XI=IX
      XL=LX
      I=(LX-IX+1)/10
      XD=I
      IF(IWF .EQ. 2) THEN
         YI=IY
         YL=LY
         I=(LY-IY+1)/10
      ELSE
         YI=(IY-1)*IZ
         YL=(LY-1)*IZ
         IY1=YI
         IY2=YL
         I=(IY1-IY2)/10
      ENDIF
      YD=I
C
C READ IN DATA FROM SELECTED FILE
C
      WRITE(IWR,510) FILTYP(IWF)
510   FORMAT(2X,'READING IN ',A4,' FILE')
      INDEX(IXSIZE)=IYSIZE*IXSIZE
      DO 512 I=IXSIZE-1,1,-1
         INDEX(I)=INDEX(I+1)-1
512   CONTINUE
      DO 520 I=IX,LX
         READ(LUS(IWF),REC=I) (VBUF(J),J=1,NWPR)
         IXYW=INDEX(I-IX+1)
         DO 514 J=IY,LY
```

```
            PBUF(IXYW)=VBUF(J)
            IXYW=IXYW-IXSIZE
514      CONTINUE
520   CONTINUE
C
C FIND MINIMUM AND MAXIMUM VALUES WITHIN THE AREA OF INTEREST
C
      VMIN=PBUF(1)
      VMAX=VMIN
      DO 530 I=1,INDEX(IXSIZE)
         IF(PBUF(I) .LT. VMIN) THEN
            VMIN=PBUF(I)
         ELSE
            IF(PBUF(I) .GT. VMAX) VMAX=PBUF(I)
         ENDIF
530   CONTINUE
C
C USER DEFINES INCREMENT FOR GENERATING CONTOUR LINES
C
      WRITE(IWR,540) VMIN,VMAX
540   FORMAT(2X,'MIN & MAX VALUES FOUND WITHIN THE AREA TO BE ',
     *           'PLOTTED: ',F10.5,', ',F10.5)
      WRITE(IWR,542)
542   FORMAT(2X,'ENTER INCREMENT VALUE FOR GENERATING CONTOUR LINES')
      CALL RUI(IRD,IWR,IH,FA,IA,NVS)
      IF(FA(1) .LE. 0.) GO TO 100
      VINCR=FA(1)
C
C SET UP PLOT
C
      CALL COMPRS
      CALL PAGE(WID,HT)
      W=WID-2.
      H=HT-2.
      CALL AREA2D(W,H)
      IF(IWF .EQ. 2) THEN
         CALL XNAME('<-X POSITION ->$',100)
         CALL YNAME('<-Y POSITION ->$',100)
      ELSE
         CALL XNAME('<- ACROSS THE FRONT ->$',100)
         CALL YNAME('<- DEPTH ->$',100)
      ENDIF
      CALL HEADIN(IHFN,100,1.5,1)
      CALL GRAF(XI,XD,XL,YL,YD,YI)
      CALL BCOMON(MXVWS)
C
C MAKE CONTOURS
C
      WRITE(IWR,550)
550   FORMAT(2X,'GENERATING CONTOUR LINES')
      CALL CONMAK(PBUF,IXSIZE,IYSIZE,VINCR)
C
C DEFINE CONTOUR LINES CHARACTERISTICS
C
      CALL CONLIN(0,'SOLID','LABELS',1,10)
      CALL CONLIN(1,'DASH','NOLABELS',1,8)
C
C DRAW CONTOUR LINES
C
      WRITE(IWR,560)
560   FORMAT(2X,'DRAWING CONTOUR LINES')
      CALL CONTUR(2,'LABELS','DRAW')
      CALL ENDPL(0)
      CALL DONEPL
      GO TO 100
C
C END
C
900   CALL CLSFIL(LUSO,MXFS)
```

```
      STOP
      END
C
C*******************************************************************************
C SUBROUTINE: SELECF
C             THIS SUBROUTINE ALLOWS THE USER TL SELECT A FILE TO PLOT
C*******************************************************************************
C
      SUBROUTINE SELECF(IRD,IWR,FILTYP,MXFS,JBUF,NWFN,IWF,IH,FA,IA)
      INTEGER*4 BLANKS,EXIT,END,FILTYP(*),JBUF(*),IH(*),IA(*)
      REAL FA(*)
      DATA BLANKS/4H    /
      DATA EXIT/4HEX  /
      DATA END/4HEND /
C
C
10    IFS=0
      IWF=0
      WRITE(IWR,20)
20    FORMAT(2X,'EXTERNAL FILES STORED IN THE CONTROL FILE')
      DO 100 I=2,MXFS
         IW=(I-2)*10+21
         WRITE(IWR,30) FILTYP(I),(JBUF(J),J=IW,IW+NWFN+3)
30       FORMAT(2X,A4,'= ',6A4,T33,'LAST WRITTEN TIME: ',2A4,' ',
     *          2A4)
         IF(JBUF(I) .NE. BLANKS) IFS=IFS+1
100   CONTINUE
      WRITE(IWR,110) EXIT
110   FORMAT(2X,A4,'= EXIT FROM FILE SELECTION MODE')
      WRITE(IWR,120) END
120   FORMAT(2X,A4,'= END THE PROGRAM')
C
C
      IF(IFS .EQ. 0) THEN
         WRITE(IWR,130)
130      FORMAT(2X,'NO EXTERNAL FILE NAMES STORED IN THE CONTROL FILE')
         RETURN
      ENDIF
C
C
      WRITE(IWR,150)
150   FORMAT(2X,'SELECT A FILE TO PLOT')
      CALL RUI(IRD,IWR,IH,FA,IA,NVS)
      DO 200 IWF=2,MXFS
         IF(IH(1) .EQ.FILTYP(IWF)) GO TO 300
200   CONTINUE
      IF(IH(1) .EQ. EXIT) RETURN
      IF(IH(1) .EQ. END) THEN
         IWF=0
         RETURN
      ENDIF
      WRITE(IWR,210)
210   FORMAT(2X,'INVALID FILE SELECTION')
      GO TO 10
C
C
300   IW=(IWF-2)*10+21
      IF(JBUF(IW) .EQ. BLANKS) THEN
         WRITE(IWR,210)
         GO TO 10
      ENDIF
      RETURN
      END
C
C*******************************************************************************
C SUBROUTINE: SETPAR
C             THIS SUBROUTINE SETS PLOTTING PARAMETERS
C*******************************************************************************
C
```

```
      SUBROUTINE SETPAR(IRD,IWR,IPAR,DEFPAR,IFTYPE,IH,FA,IA,IST)
      PARAMETER (NDIRS=10)
      INTEGER*4 IDIRS(NDIRS),IPAR(*),DEFPAR(*),IH(*),IA(*)
      REAL FA(*)
      EQUIVALENCE (IV,FV)
      DATA IDIRS/4HLD  ,4HLP  ,4HEX  ,4HEND ,4HWID ,4HHT  ,4HIC  ,
     *           4HLC  ,4HIR  ,4HLR  /
C
C
      IV=DEFPAR(1)
      WID=FV
      IV=DEFPAR(2)
      HT=FV
      IC=DEFPAR(3)
      LC=DEFPAR(4)
      IR=DEFPAR(5)
      LR=DEFPAR(6)
      IV=IPAR(1)
      WIDTH=FV
      IV=IPAR(2)
      HEIGHT=FV
10    WRITE(IWR,20) IDIRS(1),IDIRS(2),IDIRS(3),IDIRS(4),
     *              IDIRS(5),WID,IDIRS(6),HT
20    FORMAT(
     *2X,A4,' = LIST SP SUBDIRECTIVES',/,
     *2X,A4,' = LIST PLOTTING PARAMETERS',/,
     *2X,A4,' = EXIT FROM SP DIRECTIVE',/,
     *2X,A4,' = END THE PROGRAM',/,
     *2X,'***** AVAILABLE PLOTTING PARAMETERS *****',/,
     *2X,A4,' = WIDTH  OF PLOT (MAXIMUM ',F5.2,' INCHES)',/,
     *2X,A4,' = HEIGHT OF PLOT (MAXIMUM ',F5.2,' INCHES)')
      WRITE(IWR,30) IDIRS(7),IFTYPE,IC,IDIRS(8),IFTYPE,LC,
     *              IDIRS(9),IFTYPE,IR,IDIRS(10),IFTYPE,LR
30    FORMAT(
     *2X,A4,' = INITIAL COLUMN OF ',A4,' FILE TO BE PLOTTED ',
     *      '(MINIMUM ',I4,')',/,
     *2X,A4,' = LAST    COLUMN OF ',A4,' FILE TO BE PLOTTED ',
     *      '(MAXIMUM ',I4,')',/,
     *2X,A4,' = INITIAL ROW    OF ',A4,' FILE TO BE PLOTTED ',
     *      '(MINIMUM ',I4,')',/,
     *2X,A4,' = LAST    ROW    OF ',A4,' FILE TO BE PLOTTED ',
     *      '(MAXIMUM ',I4,')')
C
C
50    WRITE(IWR,60)
60    FORMAT(2X,'ENTER SP SUBDIRECTIVE NAME')
      CALL RUI(IRD,IWR,IH,FA,IA,NVS)
      DO 70 IGO=1,NDIRS
         IF(IH(1) .EQ. IDIRS(IGO)) GO TO 100
70    CONTINUE
      WRITE(IWR,80)
80    FORMAT(2X,'INVALID SP SUBDIRECTIVE')
      GO TO 50
C
C
100   IF((IGO.GT.4 .AND. IGO.LE.NDIRS) .AND. NVS.EQ.0) THEN
         WRITE(IWR,110) IDIRS(IGO)
110      FORMAT(2X,'ENTER VALUE FOR PARAMETER ',A4)
         CALL RUI(IRD,IWR,IH,FA,IA,NVS)
      ENDIF
      GO TO (10,200,900,900,300,320,400,400,500,500),IGO
C
C LP
C
200   WRITE(IWR,210) IDIRS(5),WIDTH,IDIRS(6),HEIGHT,
     *               IDIRS(7),IPAR(3),IDIRS(8),IPAR(4),
     *               IDIRS(9),IPAR(5),IDIRS(10),IPAR(6)
210   FORMAT(
     *2X,A4,' = ',F5.2,T31,A4,' = ',F5.2,/,
```

```
     *2X,A4,' = ',I4,T31,A4,' = ',I4,/,
     *2X,A4,' = ',I4,T31,A4,' = ',I4)
      GO TO 50
C
C WID
C
300   IF(FA(1).LT.5. .OR. FA(1).GT.WID) THEN
         WRITE(IWR,312) IDIRS(IGO),WID
312      FORMAT(2X,'PARAMETER ',A4,' CANNOT BE < 5.00 OR > ',F5.2)
      ELSE
         FV=FA(1)
         IPAR(IGO-4)=IV
         WIDTH=FV
      ENDIF
      GO TO 50
C
C HT
C
320   IF(FA(1).LT.5 .OR. FA(1).GT.HT) THEN
         WRITE(IWR,312) IDIRS(IGO),HT
      ELSE
         FV=FA(1)
         IPAR(IGO-4)=IV
         HEIGHT=FV
      ENDIF
      GO TO 50
C
C IC,LC
C
400   IF(IA(1).LT.IC .OR. IA(1).GT.LC) THEN
         WRITE(IWR,412) IDIRS(IGO),IC,LC
412      FORMAT(2X,'PARAMETER ',A4,' CANNOT BE < ',I4,'OR > ',I4)
      ELSE
         IPAR(IGO-4)=IA(1)
      ENDIF
      GO TO 50
C
C IR,LR
C
500   IF(IA(1).LT.IR .OR. IA(1).GT.LR) THEN
         WRITE(IWR,412) IDIRS(IGO),DEFPAR(6),DEFPAR(7)
      ELSE
         IPAR(IGO-4)=IA(1)
      ENDIF
      GO TO 50
C
C EX,END
C
900   IF(IGO .EQ. 3) THEN
         IF(IPAR(3) .GT. IPAR(4)) THEN
            WRITE(IWR,910) IDIRS(8),IDIRS(7)
910         FORMAT(2X,'PARAMETER ',A4,' IS SMALLER THAN PARAMETER ',
     *                A4,'- RESET THEM PLEASE')
            GO TO 50
         ELSE
            IF(IPAR(5) .GT. IPAR(6)) THEN
               WRITE(IWR,910) IDIRS(10),IDIRS(9)
               GO TO 50
            ENDIF
         ENDIF
         IST=0
      ELSE
         IST=-1
      ENDIF
      RETURN
      END
```

DISTRIBUTION

Naval Oceanographic and Atmospheric Research Laboratory
Stennis Space Center, MS 39529-5004

ATTN: Code 110
111
113
115 (Meert, Lundberg, Stanley, Goggins, Winchester)
125P
125L (10 Copies)
200
220
221 (Norton)
224 (Campbell)
240 (Farwell)
242 (Posey)
244 (Slater)
245 (Wagstaff)
250
300
310
311 (Ferer, Selsor)
320
321 (Pressman, Holyer, Mitchell, Smith, Lybanon)
322 (Harding, Preller, Martin, Carnes)
323 (Heburn, Blake, Hurlburt, Thompson, Kindle)
330
331 (Hollman, Saunders - 50 copies, Teague, Hallock, Boyd, Pickett, Burns)
332
333
350
351 (Walker)
352 (Mozley, Arnone)
360
362

NOARL
Code 400 (Hovermale, Tag, Brand, Haggerty, Siquig)
Monterey, CA 93943-5006

INO
Attn: (B. Willems, J. Leese, L. Kantha, E. Johnson,)

Naval Ocean Systems Center
Attn: (J. Richter Code 17)
San Diego, CA 92152

Office of Naval Technology
Attn: (CDR L. Bounds)
(Dr. M. Brisco)
(Dr. C. Votaw)
(Dr. P. Selwyn)
800 N. Quincy Street
Arlington, VA 22217-5000

Office of Naval Research
Attn: (Dr. A. Brand)
(Dr. A. Weinstein)
800 N. Quincy Street
Arlington, VA 22217-5000

Space and Naval Warfare Systems Command
Attn: Code 315, 312, PD 80
Washington, DC 20363-5000

Naval Underwater Systems Center
Attn: J. Keil, Code 60
Newport, RI 02841-5047

Naval Coastal Systems Center
Attn: D. Sheppard, Code 10
Panama City, FL 32407-5000

Naval Air Development Center
Attn: R. Becker, Code 30
Warminister, PA 18974

Naval Weapons Center
Attn: P. Arnold, Code 30
China Lake, CA 93555-6001

Naval Surface Weapons Center
Attn: A. Glazman, Code D25
White Oak
Silver Spring, MD 20910

David W. Taylor Naval Research Center
Attn: S. Goldstein Code 1203
Bethesda, MD 20084-5000

Oceanographer of the Navy
Chief of Naval Operations
Attn: OP-096 (R. Winokur, ADM Pittenger)
U.S. Naval Observatory
34th & Mass Ave, NW
Washington, DC 20392-5100

Chief of Naval Operations
Department of the Navy
Attn: OP21T
OP2172
Washington, DC 20350-2000

Center for Naval Analyses
Attn: (R. Bronowitz)
4401 Ford Avenue
Alexandria, VA 22302

Superintendent
Naval Postgraduate School
Monterey, CA 93943

Commander, Naval Sea Systems Command
Naval Sea Sys Com Headquarters
Washington, DC 20362-5101

Commander
Naval Air Systems Command
Naval Air Sys Com Headquarters
Washington, DC 20361-0001

Asst Secretary of the Navy
(Research, Engineering & Systems)
Navy Dept
Washington, DC 20350-2000

Johns Hopkins University
Applied Physics Laboratory
John Hopkins Road
Laurel, MD 20707

SAIC
Attn: (Dr. D. Rubenstein)
P.O. Box 1303
1710 Goodridge Dr.
McLean, VA 22102

Sverdrup Technology
Attn: (Dr. Ransford)
Stennis Space Center, MS 39529

Admiralty Research Establishment
Attn: (Dr. John Scott)
Southwell, Portland
Dorset DT5-2JS
United Kingdom

SACLANT Research Center
Attn: (Dr. Henry T. Perkins)
400 Via San Bartolomeo
19026 La Spezia
Italy

Oregon State University
Attn: (Dr. Robert Miller)
College of Oceanography
Oceanography Admin. Bldg. 104
Corvallis, OR 97331

REPORT DOCUMENTATION PAGE

Form Approved
OBM No. 0704-0188

1. Agency Use Only (Leave blank).

2. Report Date.
May 1991

3. Report Type and Dates Covered.
Final

4. Title and Subtitle.
Ocean Simulation Model - Version 2
First Order Frontal Simulation

5. Funding Numbers.
Program Element No. 0602435N
Project No. 03590
Task No 0M0G
Accession No. DN250019

6. Author(s).
K. D. Saunders and C. A. Cheng*

7. Performing Organization Names(s) and Address(es).
Naval Oceanographic and Atmospheric Research Laboratory
Ocean Science Directorate
Stennis Space Center, Mississippi 39529-5004

8. Performing Organization Report Number.
NOARL Technical Note 127

9. Sponsoring/Monitoring Agency Name(s) and Address(es).
Naval Oceanographic and Atmopheric Research Laboratory
Atmospheric Directorate
Monterey, CA 93943-5006

10. Sponsoring/Monitoring Agency Report Number.
NOARL Technical Note 127

11. Supplementary Notes.
*Sverdrup Technology Corporation

12a. Distribution/Availability Statement.
Approved for public release; distribution is unlimited.

12b. Distribution Code.

13. Abstract (Maximum 200 words).

Temperature, salinity, density, and sound velocity are the properties of most interest to the physical oceanographer and acoustician. The Naval Oceanographic and Atmospheric Research Laboratory has produced a first-level numerical simulation model that can produce simulated sections of temperature, salinity, density, and sound velocity in the vicinity of an oceanic front. The user controls the definition of the front.

This technical note documents the algorithms used in the simulation model and provides a users guide to the programs. Two programs are documented. The first program generates the front, and the second produces plots of the frontal properties.

14. Subject Terms.
Dynamical Oceanography, Physical Oceanography, Ocean Simulation, Modeling

15. Number of Pages.
61

16. Price Code.

17. Security Classification of Report.	18. Security Classification of This Page.	19. Security Classification of Abstract.	20. Limitation of Abstract.
Unclassified	Unclassified	Unclassified	SAR

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102